METHODS AND ALGORITHMS FOR OPTIMAL CONTROL OF FED-BATCH FERMENTATION PROCESSES

By

HAISONG CHEN

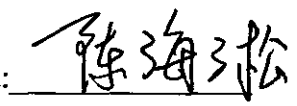
A DISSERTATION PRESENTED TO THE HIGHER DEGREES COMMITTEE OF CAPE PENINSULA UNVERSITY OF TECHNOLOGY IN FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF TECHNOLOGY (ELECTRICAL ENGINEERING)

CAPE PENINSULA UNIVERSITY OF TECHNOLOGY

2005

DECLARATION

I, Haisong Chen, hereby declare that this dissertation represents my own work and the opinions contained are my own and not necessarily those of the university. All references used have been accurately reported.

Signed:  Name: Haisong Chen

SUMMARY

Fermentation is the process that results in the formation of alcohol or organic acids on the basis of growth of bacteria, moulds or fungi on different nutritional media (Ahmed et al., 1982). Fermentation process have three modes of operation i.e. batch, fed-batch and continuous ones. The process that interests a lot of control engineers is the fed-batch fermentation process (Johnson, 1989). The Fed-batch process for the production of yeast is considered in the study.

The fermentation is based on the Saccharomyces cerevisiae yeast. It grows in both aerobic and anaerobic environmental conditions with maximum product in the aerobic conditions, also at high concentration of glucose (Njodzi, 2001). Complexity of fed-batch fermentation process, non-linearity, time varying characteristics, application of conventional analogue controllers provides poor control due to problems in tuning individual loops and the process characteristics. The problem for control of the fed-batch process for the production of yeast is further complicated by the lack of on-line sensors, lack of adequate models as a result of poorly understood dynamics. The lack of on-line sensors results in the impossibility of tuning the analogue controllers in real time.

The process for propagation of yeast in aerobic conditions is considered in the dissertation. The experiments are conducted at the University of Cape Town (UCT), Department of Chemical Engineering with a bioreactor and bio-controller combined in a Biostat ® C lab scale plant (B. Braun Biotech International, 1996).

The bio-controller has built in PID controller loops for control variables, with the ability to adjust the controller parameters i.e. P, D and I through the serial interface (Seidler, 1996). Even though the used lab scale bio-controller has the ability to monitor certain variables, the automation of the industrial bioreactors is still developing slowly

(Dochan and Bastin, 1990) with major problems experienced in modelling and measuring important control variables on-line. This existing situation is due to the characteristics of the fermentation processes as an object of control with highly non-linear, non-stationary and slow dynamics and complex relationships between the process variables. The existing control strategies in industry are based only of local PID control of some easy for measuring variables. No computer systems for monitoring and optimisation of the process (Morari and Stephanopoulos, 1980) are used.

The dissertation is overcoming the mentioned above drawbacks by developing methods, algorithms and programmes for modeling and optimal control calculation in the frame works of the developed in the department two layer system for optimal control of the Biostat ® C pilot plant with the following subsystems:

- → Data acquisition,
- → Modelling and simulation,
- → Model parameter estimation,
- → Process optimisation,
- → PID controller parameter tuning,

The dissertation formulates and solves the problems for modeling and simulation and process optimization.

The developments in the dissertation are described as follows:

Chapter 1 describes the necessity of the research discussed in the dissertation and highlights comparison between the different approaches for modelling and control of fed-batch processes for the production of yeast, (Johansson, 1993). The aim and the objectives of the dissertation are stated and explained.

Chapter 2 describes the process as an object of control looking precisely at the influence of the physiochemical variables on the biological variables. The relationship is

identified through the enzymes. The results from previous experiments are discussed to illustrate the constraints associated with the control of the process under study.

Chapter 3 describes the different types of models as applicable to the dissertation. The comparison between different types is highlighted. The derivation of the developed yeast model using mass balance equations and rate laws is discussed and presented in the chapter. The problem for simulation of the model is solved using Matlab/Simulink programs.

In Chapter 4 the optimal control problem is formulated and solved using optimal control theory, the approach of the functional of Lagrange is used. The optimisation layer problems are determined and based on the solutions of the previous upper layer i.e. the model parameters from the adaptation layer. The optimal operation of the process or yield of the yeast is based on some criteria for the production of biomass, and some constraints over minimal and maximal values of the variables. Decomposition method to solve the optimal control problem is developed on the bases of an augmented functional of Lagrange and decomposition in time domain. Algorithm of the method and program for sequential calculation in Matlab are developed.

Chapter 5 describes the concepts and evolution of parallel and distributed computing. The distributed computing toolbox and MATLAB distributed computing Engine are introduced. A parallel version of the described in chapter 4 method is developed. The building of the distributed computing programmes is described.

Chapter 6 describes the programmes of optimization in MATLAB distributed computing Engine and Simulink. The Results from calculations with the Matlab sequential program, the simulink program and the parallel calculations also are described. The results from the sequential and parallel computing are compared.

Chapter 7 presents the conclusion highlighting the developments in the dissertation as well as the future work on the topic and the possible application of the developed work in industry on the bigger scale fermentors.

The positive characteristics of the developed methods algorithms and programmes are:

- → The developed model incorporates the physiochemical variables in the biological mass balance equations. In this way: the influence of the enzymes over the biological variables is utilised and possibilities for process optimisation is created.
- → The process can be optimised in both physiochemical and biological variables.
- → The physiochemical variables can be used as control inputs to reach the process optimization.
- → The proposed method for solution of the problem of optimal control introduces new coordinating vector for time domain decomposition of the problem. In this way the complexity of the problem is reduced and the solution of the nonlinear two-point boundary value problem is avoided.
- → The introduced decomposition allows naturally application of the parallel computing cluster of computers by which the sub-problems on the optimal control problem are solved in parallel.
- → The parallel solution is for shorter time in comparison with the sequential one.
- → The optimal control of the process can be achieved without using expensive on-line sensors for measurement of the biological variables. This is why the developed system is applicable to the existing hardware and software control and measurement systems in industry.

→ The control system automates the operation of the lab scale fermentation unit. It is safe, stable and operational.

"The dissertation is a dedication to my parents"

Wenshan Chen, Lifu Qiu

ACKNOWLEDGEMENTS

The research would not have been a success without the help and the support of the following people and institutions through out the duration of the study.

Professor R. Tzoneva, my supervisor who made me believe that we can do it, through the encouragement and support that she gave me during the worst time of the study. National Research Foundation is granting the project (GUN 2052418) and funding the project. Njodzi Zizhou, for allowing us access to the experiments and giving all the help that he could. Finally and far most family, friends, students and staff of the Department of Electrical engineering at Bellville campus in Cape Peninsula University of Technology for making the environment conducive for the completion of my dissertation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Nomenclature and Glossary

D	Dilution rate
F	Volumetric feed flow rate (l/hr)
F_i	Input volumetric flow rate (l/hr)
F_o	Output volumetric flow rate (l/hr)
K_s	Monod constant or simply saturation constant
K_d	Specific death rate (hr^{-1})
m_s	Maintenance co-efficient
p	Product concentration (g/l)
p_o	Initial product concentration (g/l)
s	Substrate concentration (g/l)
s_i	Initial substrate concentration (g/l)
s_o	Reactor substrate concentration (g/l)
μ	Specific growth rate
μ_{max}	Maximum specific growth rate
V	Culture volume in the reactor vessel (litres)
V_o	Initial culture volume in the reactor vessel (litres)
x	Biomass concentration (g/l)
x_o	Initial biomass concentration (g/l)
$Y_{x/s}$	Production co-efficient biomass from substrate
$Y_{p/x}$	Production co-efficient product from substrate
L	Functional of Lagrange
λ	Conjugate variables of the functional of Lagrange,

DAQ	Data Acquisition
LabVIEW	Laboratory Virtual Instrument Engineering Workbench
"G"	Graphical Programming Language
MATLAB	Matrix Laboratory
ADI	Applikon Dependable Instruments
RS232	Recommended Standard 232.
Modelling	Process of Constructing a Mathematical Model

Adaptation	Layer in which Modelling is done
Estimation	Prediction
Mass Balance	Continuous Equations describing the balance of masses in the fermentor
Optimal Control	The best according to some criterion and constraints control
LaGrange Functional	Methodology for solution of the optimal control process
Microbial Culture	Reaction mixture
Metabolites	Products
Tuning	Calibrating for precision
Algorithm	Normally used as basis for writing a computer program. This is a set of fine rules with finite number of steps for solving a problem.
DCT	Distributed Computing Toolbox

Chapter 1

The problem for optimal control of fermentation processes, aim and objectives of the research work

1. Introduction

This chapter describes the necessity of the research discussed in the dissertation and highlights the comparison between the different approaches for modelling and control of fed-batch processes for the production of yeast. The aim and the objectives of the dissertation are stated and explained.

1.1 Awareness of the problem

Fermentation processes are widely used in SA for production of food, beverages, medicines, chemicals. Fermentation process is classified according to the mode that has been chosen for process operation that is batch, fed-batch or continuous (AGRAWAL *et al.*, 1989), (CLAES and VAN IMPE , 1998). It must be clear that the pursuit to maximize the productivity whilst minimizing the production costs is not the only reason for the control engineer's interest in fed-batch fermentation processes. Existing control system for the fermentation processes in the scientific laboratories and industry are based only on the principles of classic control. (HIGGIN, 1984), (FLAUS *et al.*, 1989). Only PID or on/off control is used. The process behavior is not optimized according to existing environmental conditions and the acting disturbances. (Ilkova and Tzonkov, 2004). The processes are not only important and industry challenging. The non-stationary process dynamics, slow response, non-linearity, sudden unexplained changes and irregularities make it more interesting for researchers. The developed project is based on the latest control technology using PC with Matlab software technology. The last challenge is to integrate the whole system i.e. DAQ and real time control system with the design methods for modeling, parameter estimation and optimal control determination, in order to build a fully automatic, adaptive and robust control system (NEIL and HARVEY, 1990)(Nikolova and. Nikolov, 2004). The problem that will be considered and solved in the thesis is connected with the process

behavior optimization based on the existing dependencies between the chemical (environmental) process variables like temperature pH, Dissolved oxygen (DO) and the biological variables (concentration of biomass substrate and product) (Zhang and Lennox 2003).

1.2 Necessity of a research work in the field of optimisation and control of the fermentation process in the production of yeast

Fermentation is the intrinsic capability of the microorganism to perform complex chemical transformations upon compounds by means of the metabolic activity and the action of enzymes. These processes are performed in tanks, called bioreactors, or fermentors and are used in the production of foods, drinks, medicines, chemical products, etc. The fermentation processes have been investigated for the past decades (Pomerleau *et al.*, 1995). Although the fermentation has its own quality control systems, sudden unexplained changes occur from time to time in the characteristics of the process, which includes: fermentation time, attenuation, contamination, pH, Temperature, Dissolved Oxygen, volumetric flow rate profiles with time (Michael and Fikret, 1992). The fermentation processes are described by two groups of variables; biological: concentrations of substrate, biomass and product; and physiochemical: temperature, pH, DO_2, agitation (Michael and Fikret, 1992).

The problem for control of fed-batch fermentation process is very important these days because of population increasing and industrial developments. The fermentation process has proved to have positive effects in many fields of domestic life and industry, but the fermentation process for a production of yeast still need deeper understanding and improvement. Furthermore this process has not been studied fully yet as an object of control. It is not clear:

- → Which kind of models are convenient, for optimal control calculation
- → Which are dependencies between input and outputs,
- → Which process variables are the most significant and in which way they affect the quality of the process or its products,
- → What the optimal combination of settings for these significant variables is, according to the goals posed on the process quality and the product quantity.

→ There is a lack of methods for measuring of important variables, for modelling and parameter estimation and for optimisation and control calculation.

The study is overcoming the constraints highlighted above in the following way:

→ For the considered case the biological variables: biomass concentration [x], substrate concentration [s], and product concentration [p] are described on the basis of the systematic approach using mass balance equations. On the same token equations, kinetic parameters μ_{max}, K_s, $Y_{x/s}$ and $Y_{p/s}$ are obtained on the basis of the empirical approach with experiments and parameters estimation technique.

Unstructured model describing the behaviour of the biological variables identifying the dependencies between physiochemical and biological variables is introduced.

→ The considered input variables are:

→ Inputs used as control signals

- Base/acid flow rate
- On/off heating energy
- Flow rate of oxygen
- Flow rate of substrate
- T
- pH
- DO_2 concentration.

→ Outputs and state space variables

- Concentration of biomass
- Concentration of substrate
- Concentration of product
- Volume of the reactor.

→ The influence of each physiochemical variable is describing behaviour of the biological variables by representation of the kinetic parameters as quadratic function of these variables.

→ The optimal settings of the process variables are obtained by solution of the problem for optimal control, together with the problem for local control in a

scheme of repetitive optimisation based on real time model parameter estimation.

→ Measurement only of the physiochemical variables is used for the adaptive, optimal control of the process.

→ New method for optimal control is developed.

1.3 Comparative analysis of the approaches for modelling and control of fed-batch fermentation process

The requirements for implementation of the model and the control of the fed-batch process are evident on the basis of the existing situation in industry. They can be used as criteria for comparison of the existing methods for modelling and control, as follow (Luyben, 1990):

→ To use simple model, convenient for calculations in real time,

→ To use simple methods for modelling and control, which do not require complex calculation and more expensive hardware.

→ Complex sensors for measurements of biological variables not to be used,

→ Control to be based on the available hardware in industry to minimize costs.

→ The hardware and software system to be simple and user friendly as possible to allow easy dialog with the operator.

Modelling

For a long time, the modelling of the dependence of the specific growth rate in microbial growth systems with respect to the process components and the physiochemical conditions has been an active research area (Hwang et al., 1995). The modelling is based on using unstructured models. They give the most fundamental observations concerning microbial metabolic processes and can be considered a good approximation when the cell composition is time dependent or when the substrate concentration is high compared to the saturation concentration (Birol et al., 1998, Moser, 1985).

The model used for control of industry fermentor are often based on simple, unstructured models since the process computer will adjust the model parameters based on the response of the system to disturbances.

The mathematical description of the considered in the dissertation model is concerned with the basic metabolic processes of Saccharomyces cerevisiae. The possibility for glucose or ethanol inhibition on both yeast cell growth and ethanol production is considered. For ethanol fermentation, the utilisation of the carbon source cannot be split into two processes, namely production of biomass and formation of product, since ethanol is the product of energy metabolism associated directly with biomass growth. According to, the most important metabolic processes accompanying ethanol fermentation can be arranged into some fundamental types of reaction patterns:

- → Catabolic pathways (breakdown of substrates into energy and small molecules).
- → Anabolic pathways (Synthesis of precursors for biomass).
- → Polymerisation of the precursors to biomass.

Maintenance metabolism keeping the cellular machinery operative.

Optimisation

In industry the aim of the process control for production of yeast is to maximize yield but very often the profiles of the feeding substrate for maximum yield may be contradictory with that for maximum productivity. In this case in (Belgardt and Juan, 1991) and (Gou *et al.*, 1995) a general objective function is used for global process optimization. It is named profit function and it is defined as the gross profit of a process divided by the production period. For a given yeast plant, the gross profit may be evaluated by material and energy balances using some simplified metabolic model of the process. This approach requires parameter estimation, but maximizing the profit function is not based on formulation and solution of a problem for optimal control.
Determination of the feeding trajectory is done in different ways in existing literature. It can be done based on an open-loop optimization if an existing mathematical model is at disposal (Meszaros and Boles, 1992). The feeding profile can be determined also according to different stages of the fermentation process (Patnaick, 1995), (Tartanousky et al., 1995) and (Johnson, 1993). These two approaches consider that the system can be exactly translated in a set of mass balance equations. However, due to a non-identified physiological problem of the cells the specific growth rate can be either higher or lower than the one that was previously established. The open-loop

optimisation does not result in optimal operation. Adaptive and closed loop control is needed to overcome the process of non-linearity and non-stationary behaviour. The literature reveals that only in a few cases the optimisation is combined with underlying feedback regulators.

Adaptive control

Usually adaptive control is realised as continuous updating the coefficients of a linear controller, (Williams et al., 1984) based on an on-line model parameter estimation and controller coefficient calculation. In this case the non-linear plant is considered as a sequence of pseudo-linear descriptions. The drawback of the approach is that the set points are kept constant and the non-linear dynamics are not taken into account (Isidori, 1995)

Most of the papers consider only one control parameter the feeding rate of the substrate. In the dissertation the control input is extended with the values of T, pH, DO_2. This makes control more flexible and corresponding to the characteristics of the process.
Since the fermentation processes are inherently non-linear the control design must be able to deal efficiently with process variations. There are in principle three different approaches

- → Robust control design,
- → Adaptive control,
- → Non-linear control (Anderson *et al.*, 1991).

The difficulty of the robust control method is that it requires a description of the model uncertainties, which reflects only the expected variations in process behaviour, and either too conservative or too optimistic control design will result.
The adaptive approach changes the model parameters in real time and design the controller parameters in real time. The non-linear approach uses the non-linear model of the process to design a non-linear controller. It however is very sensitive to parameter variations. The best results can be obtained from an approach of combination of these three techniques. When non-linear model and non-linear linearising controllers are used there are two main approaches to cope with model uncertainties.

- → Finding proper parameterisations of the model and its estimation in real time (Bastin and Dochan, 1990).

→ The controller has to be tuned in such a way that it compensate for the errors in the feed forward path of the controller by sensitivity analysis (Claes and Van Impe, 1998a,b, Claes and Van Impe, 1999)

Feedback control of yeast production processes

Industrial fed-batch production is traditionally carried out in an open loop using precalculated substrate feeding profiles. During the last decade the improved production is obtained with feeding profiles that are calculated on-line in a feedback loop.

There are basically 3 reasons for the use of advanced control strategies in the fed-batch yeast production:

1. The conflict between yield and productivity (Njodzi, 2001).
2. The level of ethanol concentration can enhance the production of inhibitory substances.
3. The reproducibility of cultivations is an important factor for a good and or uniform quality of the yeast and is not easy to be obtained without efficient feedback control.

Numerous feedback control strategies have been described in the literature to solve these difficulties by using the substrate feed rate as the control action in order to: (Chen *et al.*, 1992)

→ To set the respiratory quotient RQ close to 1. RQ is a ratio between the carbon dioxide evolution rate (CER) and the oxygen uptake rate (OUR).

→ To maintain the glucose concentration at a constant low level.

→ To track an exponential profile for the amount of biomass.

→ To set the overall specific growth rate μ_{tot}, or OUR or CER at a constant value.

→ To keep the ethanol concentration at a constant level, or to track a given ethanol profile.

The most frequently used strategy is the first one.

The controller techniques used to realise these strategies are:

1. PID

a. Early attempts for feedback control were made with a variety of PID controllers using glucose feed rate as a control action. The performance of a classical PID controller around a pre-calculated profile F^0 can be improved by calculating this profile on-line in order to reduce the disturbance on the control action (Keulers *et al.*, 1993)

2. Linear adaptive techniques

a. The next step is development of linear adaptive control in order to cope with the non-linearity, the non-stationary behaviour and the modelling uncertainties. The controllers are based on linearization around a set point and the non-linearity is interpreted by the time varying parameters of the model, which are adapted on-line.

In these applications the parameters of the linear model are estimated on-line, in general with a recursive identification algorithm. The estimated parameters are then used to design a linear controller at each sampling time. The design of the controller is based either on a quadratic cost function or on the pole placement principle (Williams et al., 1986)

3. Adaptive non-linear control

Since fed-batch systems are non-linear, it is more interesting to exploit in the design the non-linear structure and the available physical knowledge about the system. At the same time the kinetic parameters are highly uncertain. The adaptive non-linear control technical is very convenient (Pomerlean and Viel, 1992).

The results are applied to industrial fermentor without changing the tuning parameters. The non-linear linearising controller is designed. It is used with state estimator in order to realise the control action.

Predictive control

A predictive feedback control can also be set up to form a product or grow cells (Kleman *et al.*, 1991). The control scheme is divided in two parts:

A feed forward component that predicts (according to some statistical method and the previous collected data) the "need" of a certain substrate measurable on-line, and a feedback controller, which corrects for minor errors in the predicted "need". The main advantage to this system is that the investigator does not need to know the metabolic constants for a given microorganism prior to growth of that organism in the system and

it can be applied to any substrate that can be measured on-line, being particularly valuable in the minimisation of by-products.

Real time control

In the fermentation industry computer facilities have been used only off-line for acquisition of data and some mathematical analysis. Only few companies have installed computers for process control and optimisation for fermentation plants of manufacturing scales (Williams *et al.*, 1984). By using fed-batch fermentation, industry takes advantage of the fact that the concentration of the limiting substrate may be maintained at a very low level, thus:

- → Avoiding repressive effects of high substrate concentration.
- → Controlling the organism's growth rate and consequently controlling the oxygen demand of the fermentation.

Saccharomyces cerevisiae is industrially produced using the fed-batch techniques so as to maintain the glucose at very low concentration, maximizing the biomass yield and minimizing the production of ethanol (MC Neil and Harvey, 1990), (Neway, 1990), (Standbury, *et al.*, 1993)

Control of fermentation processes operated in fed-batch mode is discussed in several studies in the literature (Jorgensen and Jensen, 1989).

Advanced monitoring and control functions are performed by using the graphical programming environment called LabVIEW (Claus and Van Impe, 1998a) in a system for adaptive control of fed-batch yeast fermentation (Claus and Van Impe, 1999).

New directions in the control of fermentation processes

The research trend towards the exploitation of new methods capable of manipulating and utilising uncertain, qualitative and informal knowledge appeared recently. They are (Shimizu and Ye, 1995):

- → Control based on expert identification of the physiology state of the cells. (Konstantino and Yoshida , 1989), (Shimizu *et al.*, 1994). The problem is to identify adequately the physiological state on-line.
- → Fuzzy control

It is applicable for biotechnological processes difficult to be described by mathematical models. Different applications are described in (Nakamura et al., 1985) (Rosimeire *et al.*, 2002), (Park *et al.*, 1993). Combination of optimal feed trajectory with fuzzy logic control is good achievement of real time process control (Jin *et al.*, 1994).

→ Neuro and neuro-fuzzy control

The artificial neural networks (ANN) have ability of knowledge acquisition focusing on the application to the identification and control of chemical and biotechnological processes. ANN can approximate large classes of non-linear functions and are capable of adjusting dynamically to environment changes. The applications are done in (Karim and Rivera, 1992), (Shi and Shimizu, 1993).

→ Genetic algorithms

They have the ability of global optimisation and are used for on-line process optimisation. (Roubos *et al.*, 2000).

Common characteristics of the process influencing its control

Yeast requires aerobic conditions to grow rapidly. Oxygen starvation leads to ethanol production with a reduction in cellular growth. An excess of nutrient will also produce ethanol even in the presence of sufficient oxygen.

The yeast production requires control of different parameters at different stages of the fermentation. Two phases can be distinguished:

→ A phase in which the substrate needs to be controlled so as to avoid by-products formation.

→ A second stage in which, due to the high all density, oxygen transfer is limiting and so this parameter is the one to be controlled above a critical value under which the cellular metabolism changes. (Bajpai and Lueke, 1990)

→ DO_2 min is introduced to overcome the oxygen starvation.

→ F_{max} is introduced to overcome the excess of nutrients.

The choice of each parameter to control the process is system dependent and the decision should be based on convenience and experimental data (Turner *et al.*, 1994)

1.4 Statement of the problem

The problem considered at the study belongs to category of development of industrial and scientific control technologies by using methods of optimal control, mathematical modeling, decomposition and computation for different type of fermentation processes (I. Rocha and Ferreira, 2002).

The problem can be stated as follows:

To develop method, algorithms and programmes for optimal control of the fed-batch fermentation processes such that the maximum concentration of the biomass is produced at the end of the process. The common problem can be decomposed in a couple of subproblems:

1.4.1. Sub-problem 1: Mathematical modeling

The design and implementation of the connection between chemical an biological variables in the existing process kinetic model

1.4.2. Sub-problem 2: Control problem formulation and solution

Development of method and algorithm for solution of the optimal control problem based on decomposition of the initial problem

1.4.3. Sub-problem 3: Software developments on one computer

Development of Matlab programs for optimal control calculation using sequential mode of calculation

1.4.4. Sub-problem 4: Simulation

Simulation of the process under the developed in p.1.4.3 optimal control to verify and test method and programmes

1.4.5. Sub-problem 5: Software development on a cluster of computers

Development of Matlab programs for optimal control calculation using distributed mode of calculation.

1.5 Proposed strategy for control

The proposed control strategy is based on the characteristics of the process as an object of control and on the aims of control implementation. The strategy is based on combination of the optimal control (Roberts and Lin, 1991) with adaptive control approaches.

1.5.1 Layers of the control system

The control system is based on adaptive control strategy where the fermentation process is optimised according to a certain criterion and optimal control is realised by the PC (Morari, and Stephanopoulos, 1980). In the considered case the controller learns about the process by acquiring data from the process and keeps on updating the control model on-line then calculates and, implement the designed models and control. The three-layer control structure is used to solve the problem for modelling and optimal control first on design stage and second on control implementation stage.

There are three layers that make the proposed control strategy feasible Fig 1.1. (Sing and Titli, 1978)

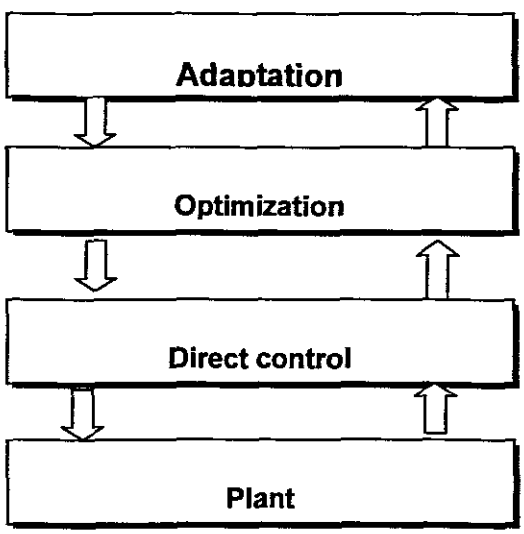


Figure 1.1: Three layers of the control structure

The problem solved on each layer is as follows:

→ Adaptation = model identification (model parameter estimation).

→ Optimisation = optimisation based on the model parameters from the adaptation layer above, and design of the direct controller parameters.

→ Direct control = realisation and implementation of the closed loop linear and non-linear control according to the parameter estimation from the adaptation layer and controller parameters and the reference trajectories from the optimisation layer.

1.5.2 Adaptation layer

Adaptive control is the name given to a control system in which the controller learns about the process by acquiring data from a certain process and keeps on updating the control model.

The latter is achieved by identifying the accurate mathematical model that represents the reaction and reactor environment. Using the control model we can progress beyond environmental control of bioreactors into the area of direct biological control.

1.5.2.1 Process modeling

Models are mathematical relationship between process variables. Traditionally the theoretical relationships provide the structure of the model. The latter is not applicable for biological process, because of the complexity of cellular processes and a larger number of environmental factors that affect cell culture.

Two types of unstructured models of the process are derived in the dissertation.

- ⇨ Model describing the behaviour of physiochemical variables.
- ⇨ Model describing the behaviour of the biological variables identifying the dependencies between physiochemical and biological variables.

1.5.2.2 Parameter estimation

The parameters of the model are determined on the bases of data from the process variables, control and state variables. Measurements and calculations are carried out on-line in real time. The obtained parameters are sent to the optimisation layer.

1.5.3 Optimization layer

The optimal control problem is solved using optimal control theory.

The optimisation layer is determined and based on the previous layer i.e. adaptation layer. The optimal operation or yield of the product is based on some criteria for the production of biomass, product, and some constraints over minimal and maximal values of the physiochemical variables. The obtained optimal trajectory of pH, temperature and DO_2 are used to determine the set points of the controllers from the direct control layer.

1.5.4 Direct control layer

This is the bottom layer. The optimal trajectory of the physiochemical variables from the optimisation layer above is passed through to this layer. On their basis and on the basis of the real measurements of the variables and the model of the physiochemical variables the optimal tunings of the controller parameters is calculated. Then the optimal trajectories and optimal settings are sent to the local controllers for the optimal control realisation.

The overall scheme for on-line implementation of the adaptive control strategy is presented in Fig1.2.

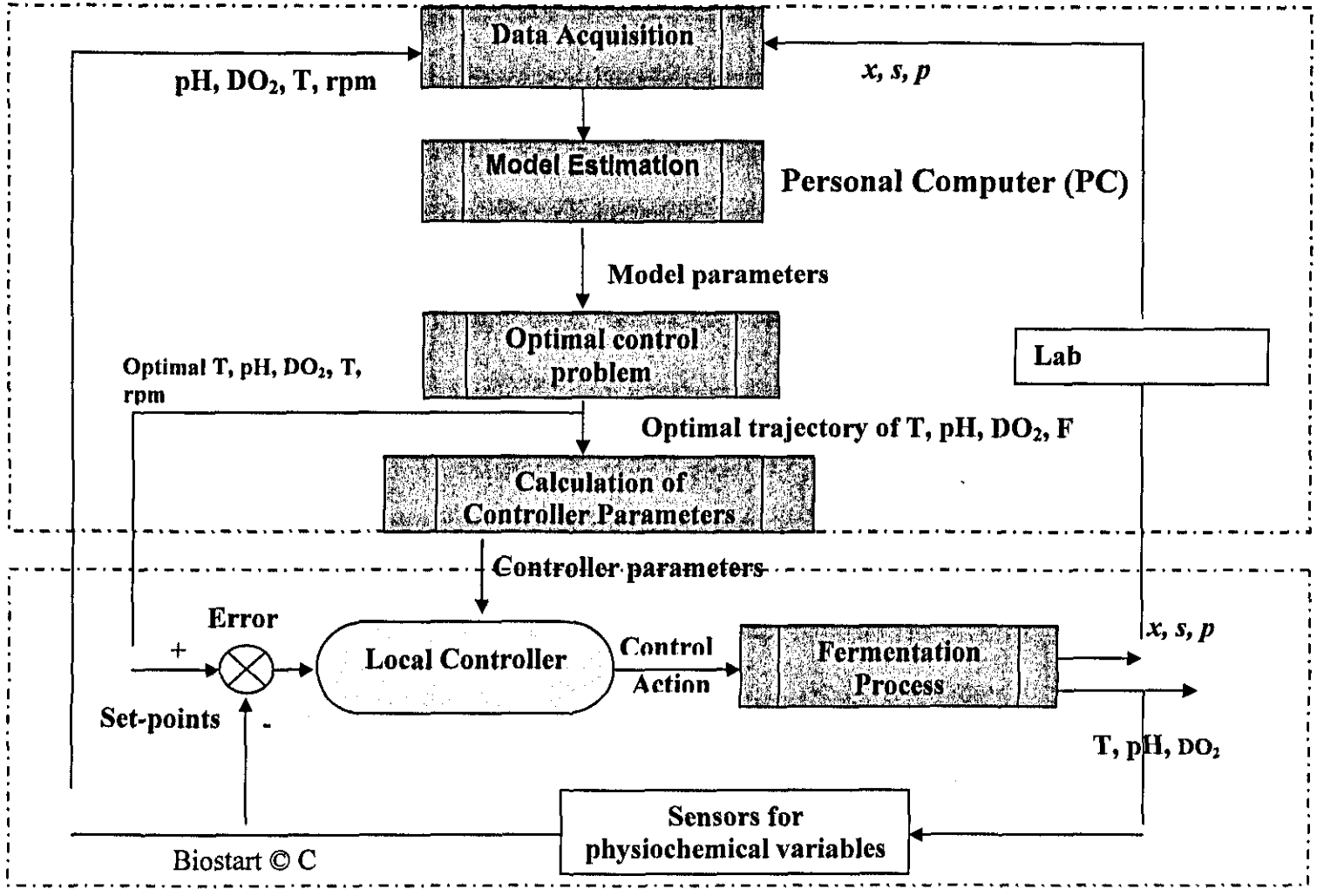


Figure 1.2: Control structure

The work of the scheme is based on the development of data acquisition system, algorithms and programs organising implementation of the programs for layer's problems and connection between layers.

1.6 Hypothesis

The hypothesis is connected with the possibilities for application of the methods and technologies of the modern control to the fed-batch fermentation processes. It can be formulated as:

— the connection between the chemical and biological variables is possibly to be created using special representation of the kinetic model parameters.

— the created model can describe very well the process behavior and the optimal control problem formulated and solved on its basis will optimize both chemical and biological process variables.

— the parallel computation of the optimal control problem will reduce the time for computation.

1.7 Delimitation of Research

1.7.1 The research study is conducted at the Cape Peninsula University of Technology in Department of Electrical Engineering and in the Department of Chemical Engineering , UCT.

1.7.2 The research is concentrate only on fed-batch processes. Other types processes: batch and continuous will not be considered.

1.7.3 The monitoring of the process will be on the bases of measurement of temperature, DO, pH and rpm by the bio-controller and sensors connected to PC.

1.7.4 The problem for optimal control is solved using only decomposition approach. The theory of the variational calculus with is used, not other methods for optimal control (principle of maximum or dynamic programming).

1.8 Motivation of the research

The result of the control will be received if the control system succeeds to optimize the environmental conditions for the cells. It is therefore quite important to investigate how the chemical state cause changes in biological state. In order to achieve the full biological potential, the chemical or environment conditions must be maintained optimally. The chemical variables control the enzyme activity of the biological variables and consequently control the biological. Therefore, they are considered as control inputs. This means that if the optimal conditions for the chemical variables are reached according to a given optimization criterion, then the optimal conditions for the biological will be also reached. The idea is not new but there has never been any practical industrial application of this method.

The ability to control a fed-batch fermentation process accurately by following desired trajectories is believed to be a key factor in maintaining the product consistency and this believe can be proved by the undertaken research project.

1.9 Assumption

The assumptions are made according to different parts of the project research work, as follows:

- Process under study → it is working according to prescribed technology.
- Model of the process → the kinetic model of the fed-batch process can be used as a basis for description of the connections between the chemical and biological variables.
- Optimal control problem solution → The Lagrange functionals methodology can successfully be used to solve nonlinear problems for optimal control
- Real time control implementation → the calculation algorithms PC programs, bio-controller and the whole hardware and software structure will allow quick, according to the dynamics of the process real time solution.

1.10 Objectives of the research project

The proposed project consider the potential impact of improved operation and computer controlled technologies over plant efficiency

The aim of the project is :

To develop methods algorithms and programmes for real time calculation of the optimal control of fed-batch fermentation processes by which to achieve optimal production of biomass from the fed-batch fermentation process for production of yeast

The project objectives are:

- To develop a unique mathematical model of the fermentation process. This is achieved by performing a study of various existing models and by studying the complex dependencies between the process variables
- To develop decomposition method for optimal control calculation of the yeast production process. This is achieved by performing a study of the existing methods for optimal control on the basis of functional of Lagrange.
- To develop algorithms and programs in Matlab for sequential calculation of the optimal control. The functions from Matlab, capable to perform some of the parts of the calculation algorithm are used.
- To develop an algorithm and a program in Matlab for distributed calculation of the optimal control. The special version of Matlab for parallel computing is studied and implemented in the existing in the Department cluster for High performance computing.

1.11 Conclusion

This chapter describes the necessity of the research and highlights comparison between the different approaches for modelling and control of fed-batch processes for the production of yeast. The aim and the objectives of the dissertation are stated and explained. The selected development and simulation software used is discussed. The process as an object of control looking precisely at the influence of the physiochemical variables on the biological variables will be described in the next chapter.

Chapter 2

The fed-batch fermentation process characteristics

2. Introduction

This chapter describes the process as an object of control looking precisely at the influence of the physiochemical variables on the biological variables. The relationship is identified through the enzymes and the latter is discussed in the chapter. The results from previous experiments are discussed to illustrate the constraints associated with the control of the process under study.

2.1 Fermentation process

The fermentation process takes place in tanks called bioreactors or fermentors. Fermentation processes are complex, dynamic autocatalytic processes in which transformation, caused by biological catalysts called enzymes, takes place.

Enzymes are nothing but proteins of high molecular weight that act as catalysts. They are substrate specific, versatile, and very effective biological catalysts resulting in much higher reaction rates as compared to chemically catalysed reactions under ambient temperature. Enzymes are classified according to the reaction they catalyse. Enzymes decrease the activation energy {E} of the reaction catalysed by binding the substrate and forming an enzyme-substrate {ES} complex. Molecular aspects of enzymes-substrate interaction are not yet fully understood, but yet the enzymes require optimal conditions for pH, temperature, ionic strength, etc for their maximum activity. (Michael and Fikret, 1992).

The technology of the fermentation is as follows:

The nutrients are added and the cells are inoculated into the fermentation vessel, environmental conditions such as pH, temperature, DO_2, etc are kept convenient. The cells starts multiplying forming mass cells called biomass, and form other substances called

secondary metabolites, or products. There are three different modes of the fermentation processes.

⇨ **Batch fermentation:**

The input (feed-nutrients) to the process is added during the initial stage of the process and the product (process output) is not removed until the process is finished. The main disadvantage of the batch process is that it is characterised by high down time between batches, comprising the charge and discharge of the fermentor vessel, the cleaning, sterilisation and the re-start of the whole process.

⇨ **Fed-batch fermentation:**

The input to the process is continuously added from the initial stage through the end of the process, but the process output is not removed until the process is finished.

⇨ **Continuous fermentation:**

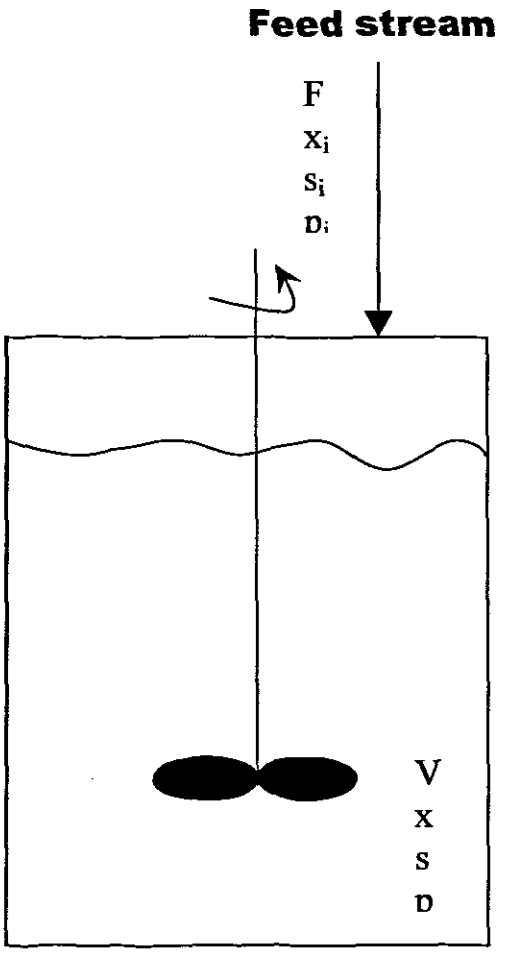
The input to the process is continuously added and the process output is continuously removed making it more economic compared to the other two type mentioned above.

2.2. Fed-batch fermentation process

The three processes mentioned above have their advantages and disadvantages, but the one that interests most of the control engineers is the *Fed-batch fermentation process*. Yeast producers originally devised the fed-batch technique in the early 1900s to regulate the growth in batch culture of *Saccharomyces cerevisiae.*

The fed-batch processes start with the cells being grown under the batch movement for some time, usually until close to the end of the exponential growth phase. At this point solution of substrate (nutrients) is feed into the reactor, without the removal of culture fluids. This feed should be balanced enough to keep the growth of the microorganisms at a desired specific growth rate and reducing simultaneously the production of by-products. During the growth phase nutrients are added (Flow rate - F) when needed and acid and base is added to control the level of pH. If the cells are grown in an aerobic condition, using appropriate meters the oxygen flow rate can be measured. Also the rate into which the oxygen is supplied can be controlled. Temperature is also one of the variables that need to

be controlled precisely. There are many variables that influence the operation of the process. The ones listed above are just the few that can be measured on-line. The diagram of the fed-batch process is given on Fig 2.1.



F ⇒ Feed rate for nutrient substrate in $[m^3/s]$,

x_i ⇒ Biomass concentration input flow in $[mg/l]$,

s_i ⇒ Substrate concentration input at initial moment $s_i(0)=s_i$=constant in $[mg/l]$,

p_i ⇒ product concentration in the input flow, at initial moment $p(0)=0$ in $[mg/l]$,

V ⇒ Culture volume in the reactor vessel, at initial moment $V(0)=V_0$ in $[l]$,

x $\Rightarrow$ Biomass concentration, at initial moment $x(0)=x_0$ in $[mg/l]$,

s $\Rightarrow$ Substrate concentration, at initial moment $s(0)=s_0$ in $[mg/l]$,

p $\Rightarrow$ Product concentration, in the fermentor $[mg/l]$

Figure 2.1 Diagram for a stirred fed-batch fermentor

2.3 Fed-batch fermentation for production of yeast

Yeast is widely used in brewing, baking, wine industry, research etc. The optimal operation of the processes discussed above depends on the efficient inoculum's development stage. The dissertation is considering the growth of yeast cells using the fed-batch fermentation process. The study is looking at the optimal operation of the fermentation of yeast used to inoculate fermentation of beer. The aim is to maximise the production of yeast. Yeast grows in two different cultures, presence of oxygen (Aerobic culture) and absence of oxygen (Anaerobic culture).

2.3.1 Aerobic fed-batch culture

The fed-batch operation requires the identification of the control variables and their respective limits. The oxygen availability have a great impact on the biomass, even though it is inoculated in excess it may inhibit the growth rate.

2.3.2 Anaerobic fed-batch culture

Saccharomyces cerevisiae is able to grow in the absence of oxygen i.e. anaerobic fed-batch culture. The maximum specific growth is lower in anaerobic growth (Njodzi, 2001). The latter resulted in the anaerobic fed batch culture not to be considered in this case. Anaerobic propagation is not ideal for inoculums development and is used for fermentation processes for production of beer.

2.3.3 Variables characterizing growth kinetics of yeast.

The fermentation process is characterised by three key groups of variables Table 3.1. Most of the physiochemical variables can be measured very easy in real time with the available sensors. In contrast biological variables cannot be measured on-line with the sensors that

are available. The investigation and experiments illustrates that physiological variables affect biological variables in a certain way, and by identifying the optimal conditions of the physiological variables, conclusion about the optimal conditions for biological variables can be made through the enzymes link (Dochan, 1990).

Physical	Chemical	Biological
⇨ Temperature	PH	Biomass concentration
⇨ Pressure	Dissolved O_2 (DO_2)	Enzyme concentration
⇨ Reactor weight	Dissolved CO_2	Biomass composition
⇨ Liquid level	Redox potential	(such as DNA, RNA,
⇨ Foam level	Exit gas composition	Protein, ATP/ADP/AMP,
⇨ Agitator speed	Conductivity	NAD/NADH levels)
⇨ Power consumption	Broth composition	Viability
⇨ Gas flow rate	(substrate, product, ion	Morphology
⇨ Medium flow rate	Concentration, etc.)	
⇨ Culture viscosity		
⇨ Gas hold-up		

Table 2.1 Variables measured or controlled in bioreactors for yeast production.

The table above illustrates clearly that there is a lot of factors that affect the growth of the yeast cells. The study does not consider all the variables mentioned in table 3.1 as the latter may lead into the process model being complex and impossible to implemented in real time with the available software and hardware. For the purpose of the dissertation and from the control point of view the following notation and variables are used in the dissertation.

- → Physiochemical variables used for influencing the process: Temperature (T), pH, Dissolved Oxygen (DO_2) and Flow rate (F).
- → Biological variables that is biomass (x), substrate (s), and product (p) concentrations.

The substrate and product are considered in connection to the biomass and because of that they are considered as biological variables.

2.3.3.1 The influence of temperature

Growth of yeast cells is affected by fluctuations in temperature. The latter implies that an optimal fermentation temperature has to be identified. Optimal temperatures are classified according to the groups of the microorganisms to which they are applied as

- → (T_{opt} < 20°C), Psychrophiles,
- → (T_{opt} = between 20°C to 50°C), Mesophiles, and
- → (T_{opt} => 50°C), Thermophiles.

Looking at temperature against growth rate, it is concluded that temperature can influence biological process by kinetic effects on reaction rate and catalytic effects on the activity of enzymes.

As the temperature increases towards the optimal temperature T_{opt}, μ the growth rate approximately doubles for every 10°C increase in temperature.

Above the optimal temperature range, the growth rate decreases and the thermal death may occur due to high temperatures.

Net growth rate can be expressed above the optimal temperature as follows:

$$\frac{dx(t)}{dt} = (\mu - k_d)x \qquad (2.1)$$

Where μ is the growth rate and k_d is the death rate.

At high temperatures, the thermal death rate exceeds the growth rate, which result in a decrease in the concentration of viable cells.

Both μ and k_d vary with temperature according to the Arrehenius equations

$$\mu = Ae^{-Ea/RT}, kd = A'e^{-E_d/RT} \qquad (2.2)$$

Where E_a and E_d are activation energy for growth and thermal death. Activation energy for growth rate is typically 10 to 20 [kcal/mol] and for thermal death, 60 to 80 [kcal/mol]

implying that the thermal death is more sensitive to temperature changes compared with microbial growth.

Looking at temperature against product formed, the optimal temperature for growth and product formation may be different. Growth rate and product formation are not the only ones affected by the temperature but the yield co-efficient $Y_{x/s}$ is also affected. As with the product formation when the temperature increases above the optimal temperature, the maintenance requirements of the cell increases. That is, the maintenance coefficient increases with increasing temperature with activation energy of 15 to 20 [kcal/mol], resulting in a decrease in the yield coefficient.

2.3.3.2 The influence of pH

Industry in general has a tendency of keeping some of controllable variables constant. pH being one of those variables.

Hydrogen ion concentration (pH) affects the activity of the enzymes and therefore the microbial growth rate. Same with the temperature the optimal pH for growth rate may be different from that for product formation. Different organisms have different pH optimal values and they vary about the optimum by +-1 to 2 pH units.

Yeast optimal pH varies between 3 and 6. Note, when the pH differs from the optimal value, the maintenance energy requirements increases.

Nitrogen source affects the increase or decrease in pH level. The latter is due to the fact that hydrogen ions are either released or removed from the medium, resulting in pH variations. Also, pH can change because of the production of organic acids, the utilization of acids or the production of bases. Thus the control and monitoring of pH is very important.

2.3.3.3 The influence from dissolved oxygen (DO_2)

Microorganisms need oxygen to develop properly (Njodzi, 2001). Saccharomyces C. can grow in the presence or absence of oxygen that is, either in aerobic or anaerobic conditions. The latter implies that dissolved oxygen (DO_2) concentration has to be monitored and most of all controlled in the aerobic condition. This is because the concentration of DO_2 affects the final biomass yield, growth rate and other by products and may be the limiting

substrate, since oxygen is a sparingly soluble gas in water. Nutrients flow rate (F) and oxygen transfer rate (OTR) as well as other control physiochemical variables needs to be controlled and maintained at a specific optimal value such that optimal results are obtained as discussed in the paragraph above.

The dissolved oxygen is represented by the following mass balance equation in the bioreactor:

$$\frac{dC}{dt} = OTR - OUR - C \qquad (2.3)$$

Where:

C is the DO_2 concentration

DO_2 is the DO concentration in the reactor.

OTR is the oxygen transfer rate.

OUR is the oxygen uptake rate.

OUR depends on the biomass growth and is given by

$$OUR = k_3 \mu x \qquad (2.4)$$

with k_3 being yield coefficient. A term $k_r x$ is sometimes included to account for the endogenous respiration.

$$OUR = k_3 \mu x + k_r x \qquad (2.5)$$

By using a line of reasoning based on Henry's law to model the liquid gas transfer dynamics the OTR becomes:

$$OTR = k_L a(C_s - C) \qquad (2.6)$$

Where $k_L a$ = mass transfer coefficient and C_s being oxygen saturation coefficient.

There are some constraints associated with the mass balance equation. It is often useless because the coefficient C_s and k_La may be unknown and highly time varying. The literature revels that oxygen saturation concentration C_s depends on variables such as oxygen partial pressure in surrounding atmosphere, temperature, salinity and concentration of surface in the liquid and other factors such as the type and the geometry of the reactor or the airflow rate determines the value of k_La.

Industry tries to over comes the constraints above by measuring on-line input and output gaseous flow rates. Hence, if the liquid –gas transfer dynamics are negligible as often assumed, the OTR can simply be expressed from the gaseous oxygen balance:

$$OTR = Q_1\text{-}Q_2 \qquad (2.7)$$

Where Q_1-Q_2 are respectively the input and output oxygen flow rates (per volume unit). The latter has its own constraints in this study, because the sensors to measure Q_1-Q_2 are not available.

The study has overcomes the constraints highlighted above by considering on-line measurement of the dissolve oxygen concentration (DO_2). The measured DO_2 is one of the parameters of the developed model of the process.

2.3.4 Results in yeast production caused by the influence of different variables.

The observation from the different experiments showed the following:

- ➔ It has been observed that at high concentration substrate (malt) a by –product ethanol is produced (inhibition factor) while in low concentration of malt, the yeast growth is restricted (limiting factor) (Njodzi, 2001).
- ➔ The result from the production of ethanol during aerobic growth at high sugar concentration is called Crab-tree effects or overflow metabolism. The latter reduces the biomass production.
- ➔ Also oxygen is proportional to the yeast propagation process (biomass yield), on the other hand very high dissolved oxygen (DO_2) concentration may induce stress on yeast that may lead to lower biomass productivity.
- ➔ Quality of fermentation is also affected by the quality of the yeast used for inoculation.

2.4 Conclusion

Fed-batch fermentation for the production of yeast like object of control looking precisely looking at the influence of the physiochemical variables on the biological variables is presented and described in the chapter. The relationship is identified through the enzymes. The results from previous experiments are discussed to illustrate the constraints associated with the control of the process under study.

Chapter 3

Mass balance model of the fed-batch fermentation process for production of yeast

3. Introduction

This chapter describes the different types of models as applicable to the dissertation; the comparison between different types is highlighted. The derivation of the developed yeast model using mass balance equations and rate laws is discussed and presented. The problem for model simulation is solved using Matlab programs and the programs are described at the chapter 6.

3.1 Process modelling

To fulfil the adaptive control an accurate mathematical model representing the reaction and reactor environment has to be identified. The model is used as a tool for process analysis, for the design of control systems and for realization of the adaptive control. The question that rises is how to obtain a good process model. In this regard there are three fundamental issues:

- Model formulation,
- Parameter estimation and
- Model validation.

Traditionally the theoretical relationships provide the structure of the model. The latter is not applicable for a biological process, because of the complexity of cellular processes and a larger number of environmental factors that affect cell culture. As a result of these the biological models are developed on the basis of observations rather than theoretical laws of science (Doran *et al.*, 1995).

3.2 Model types

A model fundamental result is to obtain functional relationship between various process variables that explains the behaviour of the observed process. The underlying belief is that a true but unknown fundamental relationship dictated by natural laws exists between process variables. The modelling exercise is directed towards discovering this relationship. Traditionally, models are based on a combination of 'theoretical' relationships, which provide the structure of the model, and experimental observations, which provide the numerical values of the model coefficients. The above is very difficult when it comes to the biological process; this is due to the complexity of cellular process and also the number of environmental factors, which affect the cell structure.

There are different types of models, there is:

- Theoretical models
  - These models are identified using the systematic application of natural laws in order to determine the functional relationship between the process variables in the observed process.
- Empirical (Experimental) models
  - In contrast the empirical models find a way to approximate the functional relationship by some usually simple mathematical functions, using the data acquired during experiments, hence they are called *empirical models.*
- Semi-theoretical
  - These models are developed using both systematic and empirical approach. The developed model is semi theoretical in the sense that both systematic approach and empirical approach are applied in solving the problem under study.

From the three types of models described above it is evident that, the application into which the model has to be derived determines the model type that needs to be used. The availability of the process measurements also determines which model will be appropriate for the specific application. It is common to adopt the theoretical model when the process under study is well understood. On the other hand if the process is too complicated {implying not well understood} or the model equations are extremely complex, the empirical approach is the most appropriate one.

The main objectives of this section is to identify the model by finding the functional relationship highlighted above using the natural laws that exists between process variables in the observed process data.

For the considered case the biological variables x, s, and p are described on the basis of the systematic approach using mass balance equations. On the same token equation parameters μ_{max}, K_s, $Y_{x/s}$ and $Y_{p/s}$ are obtained on the basis of the empirical approach with experiments and parameters estimation technique.

Two types of models of the process are considered.

→ Model describing the behaviour of physiochemical variables.

→ Unstructured model describing the behaviour of the biological variables identifying the dependencies between physiochemical and biological variables.

3.3 Model describing the behaviour of physiochemical variables.

The first model is described by the first order transfer function with time delays.

$$G(s) = \frac{k^{ph} e^{-\varpi}}{T_{ph} s + 1} \qquad (3.1)$$

where K is the gain, τ is the time delay, T is the time constant.

The inputs to these functions are the flow rate of acid or base, heat source, on/off, flow rate of dissolved oxygen (DO_2), rate of electrical motor with their respective outputs: pH, temperature, dissolved oxygen concentration, number of turns of the stirrer in revolutions per minute (rpm). Fig 3.1.

Equation (3.1) has the same structure for all the physiochemical variables listed above, only the parameters K, τ, T are different.

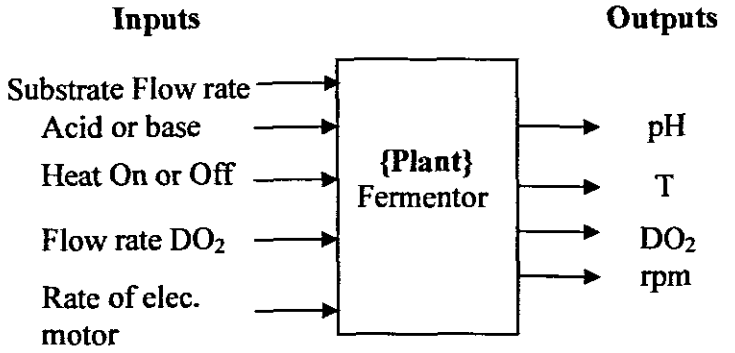


Figure 3.1 Model 1 inputs and their respective outputs

The model described above is used for optimal tuning the controllers on each physiochemical variable on the bio-controller. Parameter estimation for this model is not done in the dissertation separately as it is included as a part of the used program for auto tuning of PID controller described in chapter 4.

3.4 Model describing mass balance equations of the fed batch fermentation process

The second model is built on the basis of mass balance equations. These equations describe the dynamics of biological variables, concentrations of substrate (s), biomass (x) and product (p) and of the reactor volume (V). The equations are fundamentally derived from the law of conservation of mass that says, "Mass is conserved in ordinary chemical and physical processes". The growth rate dependencies are based on the Monod (1949) kinetics where sugar uptake is directly dependent on sugar and enzymes concentration.

The schematic diagram of the fed-batch fermentation is given on Fig 2.1.

The process starts with relatively dilute solution of substrate and more nutrients are added as the process progresses. The rate into which the nutrients are added is controlled to avoid high concentration of substrate as this may inhibit or switch on undesirable metabolic pathways (Jens *et al.*, 2002). In cultures where oxygen is used to grow the yeast cells (aerobic conditions), more oxygen is pumped to the process increasing the growth rate. From the other side high growth rate of biomass can inhibit the biomass yield by bigger amounts of produced products e.g. ethanol.

The mass balance equations are derived from the basic laws of conservation of mass represented by the following general equation:

The rate of mass in through system boundaries.	−	The rate of mass out through system boundaries.	+	The rate of mass generated within system.	−	The rate of mass consumed within system.	=	The rate of mass accumulated within system.

$$\frac{dM(t)}{dt} = \hat{M}_i(t) - \hat{M}_o(t) + R_G(t) - R_C(t) \qquad (3.2)$$

Where:

M is the mass accumulated within the system

$\hat{M}_i$ is the mass flow rate of substance "A" entering the system.

$\hat{M}_o$ is the mass flow rate of "A" leaving the system.

R_G is the mass rate of generation of species "A" by chemical reaction.

R_C is the mass rate of consumption "A" by reaction.

The dimensions of $\hat{M}_i$, $\hat{M}_o$, R_G and R_C are mass per unit time with units such as gs^{-1}, kgh^{-1}, $lbmin^{-1}$ etc. All the variables can vary with time.

Using mass balance equations the theoretical model representing the concentration of biomass yield, product and substrate is described. The equations are obtained sequentially for the biological variables i.e. biomass, substrate and product.

3.4.1 Expressions for growth and production rates

3.4.1.1 Growth rate laws

The letter t used for time dependence is missing in this part, but all variables are time dependent. The common expression for the cell growth in the batch reactor is given by the equation

Cells + substrate = more cells + product (3.3)

The growth of cells in this equation can be expressed mathematically by the cell growth rate with which the new cells are formed.

$$r_g = \mu x \qquad (3.4)$$

Where r_g is the cell growth, $g/dm^3.s$, x is the cell concentration g/dm^3; μ is the specific growth rate.

The most commonly used expression for the exponential growth of microorganisms is Monod's equation. The specific growth rate can be expressed as:

$$\mu = \mu_{max} \frac{s}{K_s + s} \qquad (3.5)$$

Where μ_{max} is the maximum specific growth rate [s^{-1}], and K_s is called Monod constant or simply the substrate saturation constant. The importance of K_s is that, when substrate concentration is numerically equal to K_s, growth rate is exactly half of the maximum growth rate. For a number of bacteria and fungi the constant K_s is small and $r_g = \mu_{max}$.

In common case:

$$r_g = \frac{\mu_{max} xs}{k_s + s} \qquad (3.6)$$

In many systems the product inhibits the rate of growth.

The equation, which is used to represent the product inhibition on the growth rate, is: (Dochan et al., 1990)

$$r_g = k_{obs} \frac{\mu_{max} xs}{k_s + s} = (1 - \frac{p}{p^*})^n \frac{\mu_{max} xs}{k_s + s} \qquad (3.7)$$

Where $k_{obs} = (1 - \frac{p}{p^*})^n$ (3.8)

p^* is the product concentration at which all metabolism cease, [mg/l], n is the empirical constant for Saccharomyces cerevicerae.

Other used equations for the description of cell growth are:

→ Tessier equation

$$r_g = \mu_{max}(1 - \exp(\frac{s}{k})x) \qquad (3.9)$$

→ Moser equation

$$r_g = \frac{\mu_{max} x}{(1 + k_s \exp(-\eta))} \qquad (3.10)$$

Where η and K_s are empirical constant determined by a best fit of the data. These equations have been found to better-fit experimental data at the beginning or at the end of fermentation.

The cell death rate is given by

$$r_d = k_d x \qquad (3.11)$$

3.4.1.2 Stoichiometry

The stoichiometry for cell growth is very complex and varies with micro organism, nutrient system and environment conditions such as pH, Temperature, redox potential etc. The considerations are for cell growth, limited by only one nutrient in the medium. The equation (3.3) can be written in the following way

$$s \xrightarrow{cells} Y_{x/s}x + Y_{p/s}p \tag{3.12}$$

where the yield coefficient for the growth of biomass is:

$$Y_{x/s} = \frac{mass\_of\_new\_cells\_formed}{mass\_of\_substrate\_consumed\_to\_produce\_new\_cells} \tag{3.13}$$

The stoicheiometric yield coefficient that relates the amount of product formed per mass of substrate consumed is:

$$Y_{p/s} = \frac{mass\_of\_product\_formed}{mass\_of\_substrate\_consumed\_to\_form\_product} \tag{3.14}$$

In addition to consuming substrate to produce new cells, part of the substrate must be just to maintain a cell's daily activities. The corresponding maintenance utilization term is:

$$m = \frac{mass\_of\_substrate\_consumed\_for\_maintenance}{mass\_of\_cells\_*\_time} \tag{3.15}$$

The rate of substrate consumption for maintenance whether or not cells are growing is:

$$r_{sm} = mx \tag{3.16}$$

Product formation can take place during different phases of cell growth. When product is only produced during the growth phase then the rate of product formation is:

$$r_p = Y_{p/x} r_g \qquad (3.17)$$

However when the product is produced during the stationary phase, then the product formation is related to substrate consumption by:

$$r_p = Y_{p/s}(-r_s) \qquad (3.18)$$

The equation which relate the rate of nutrient consumption, $-r_s$, to the rates of cell growth, product generation and cell maintenance is:

Net Rate of substrate	=	Rate consumed by	+	Rate consumed to	+	Rate consumed for	(3.19)

$$-r_s = Y_{s/x} r_g + Y_{s/p} r_p + mx \qquad (3.20)$$

If the product is produced during the growth phase it may not be possible to separate out the amount of substrate consumed for growth from that consumed to produce the product. Under these circumstances all the substrate consumed is lumped into the stoichiometric coefficient, $Y_{s/x}$ and the rate of substrate disappearance is:

$$-r_s = Y_{s/x} r_g + ms \quad \rightarrow \text{product formation in the growth phase} \qquad (3.21)$$

The corresponding rate of product formation is:

$$r_p = r_g Y_{p/x} \qquad (3.22)$$

These equations cannot be used during the stationary phase. When the product is produced only during the stationary phase the nutrient consumed for growth has become virtually exhausted and a different nutrient is used for cell maintenance and to produce the desired product.

The net rate of substrate consumption during the stationary phase is:

$$-r_s = mx + \frac{Y_{sn/p} K_p s_n x}{K_{sn} + s_n} \tag{3.23}$$

where:

s_n is the concentration of the secondary nutrient [g/m^3]

K_p is the specific rate constant with product, [s^{-1}],

x is the cell concentration [g/m^3],

K_{sn} is the constant, [gm/m^3],

$$r_p = Y_{p/sn}(-r_{sn}) \tag{3.24}$$

The third situation is when the product is formed during both logarithmic and stationary phase. This is the considered case of Seccharomyces cerevicerae.

The equations for the substrate consumption is

$$-r_s = Y_{s/x} r_g + Y_{s/p} r_p + mx \tag{3.25}$$

Substituting equation (3.23) into equation (3.20) is obtained:

$$-r_s = Y_{s/x} r_g + Y_{s/p} Y_{p/x} r_g + mx \tag{3.26}$$

After grouping in equation (3.26) it is obtained:

$$-r_s = [Y_{s/x} + Y_{s/p} Y_{p/x}] r_g + mx = Y_{s/x} r_g + mx = \frac{1}{Y_{x/s}} r_g + mx \tag{3.27}$$

This common presentation in equation (3.27) shows that the rate of substrate consumption for both cell growth and product formation can be presented by one common stoichometric

coefficient. The expressions for rate laws and stoichiometry are used in mass balance equations for full description of the process behavior.

3.4.2 Biomass mass balance equation

In fed-batch operation $\hat{M}_{ox}$=0; $\hat{M}_{ix}$ is equal to the feed flow rate F multiplied by the cell concentration x_i in the feed.

$\hat{M}_{ix} = Fx_i$ = mass of cells entering the reactor.

$M_x = xV$, where x is the cell concentration and V is the liquid volume of the reactor vessel,

$R_{Gx} = \mu xV$, where: μ is the specific growth rate,

$R_{Cx} = k_d xV$, where k_d is the specific death rate constant.

Substituting in equation (3.2) above the following is obtained:

$$\frac{d(xV)}{dt} = Fx_i - 0 + \mu xV - k_d xV$$

Note V is the function of time so it cannot be cancelled out, rather a product rule for derivative can be used.

$$x\frac{dV}{dt} + V\frac{dx}{dt} = Fx_i + (\mu - k_d)xV, \text{ Substituting flow rate F with:} \quad (3.28)$$

$$F = \frac{dV}{dt} \quad (3.29)$$

$$xF + V\frac{dx}{dt} = Fx_i + (\mu - k_d)xV \quad (3.30)$$

$$\frac{dx}{dt} = \frac{F}{V}x_i + x(\mu - k_d - \frac{F}{V}) \quad (3.31)$$

Substituting dilution rate

$$D = \frac{F}{V} \quad (3.32)$$

$$\frac{dx(t)}{dt} = D(t)x_i(t) + x(t)(\mu(t) - k_d - D(t)), x(0) = x_0 \quad (3.33)$$

In fed-batch the volume increases with time, therefore if F is constant, D decreases as the reaction progresses.

Generally the feed material is sterile so that $x_i = 0$, also if cell death is negligible compared with growth so that $k_d << \mu$; the equation above becomes.

$$\frac{dx(t)}{dt} = x(t)(\mu(t) - D(t)), x(0) = x_0 \qquad (3.34)$$

3.4.3 Substrate mass balance equation

When substrate concentration is limited in the fed-batch reactor the mass balance relations for it are:

$\hat{M}_{os} = R_{Gs} = 0$

$\hat{M}_{is} =$ is the mass flow rate of substrate entering the system $= Fs_i$.

$M_s =$ is the mass of substrate in the reactor $= sV$.

$$R_{cs} = r_s V = (\frac{1}{Y^1_{x/s}} r_g + m_s)V \ , \qquad (3.35)$$

The equation describing the substrate behaviour is:

$$\frac{d(sV)}{dt} = Fs_i - (\frac{r_g}{Y^1_{x/s}} + Y_{s/p} r_p + m_s x)V \qquad (3.36)$$

$$\frac{d(sV)}{dt} = Fs_i - (\frac{r_g}{Y^1_{x/s}} + \frac{r_p}{Y_{s/p}} + m_s x)V \qquad (3.37)$$

$$\frac{d(sV)}{dt} = Fs_i - (\frac{\mu}{Y_{x/s}} + \frac{\mu}{Y_{s/p} Y_{p/s}} + m_s)xV \qquad (3.38)$$

$$\frac{d(sV)}{dt} = Fs_i - (\frac{\mu}{Y_{x/s}} + m_s)xV \qquad (3.39)$$

where:

$$Y^{1}_{x/s} = \frac{Y^{1}_{x/s}}{2}$$

$Y_{x/s}$ is the true biomass yield from the substrate.

r_p is the rate of product formation.

$Y_{p/s}$ is the true product yield from the substrate.

m_s is the maintenance co-efficient.

r_s is the volumetric rate of substrate uptake

Expanding equation (3.10) and substituting $F = \frac{dV}{dt}$ it is obtained:

$$\frac{ds}{dt} = D(s_i - s) - (Y_{s/x}\mu + Y_{s/p}\mu + m_s)x, s(0) = s_0 \qquad (3.40)$$

$$\frac{ds}{dt} = D(s_i - s) - (\frac{\mu}{Y_{x/s}} + m_s)x \qquad (3.41)$$

3.4.4 Product mass balance equation

When product concentration is considered in the fed-batch reactor the mass balance relations for it are:

$\hat{M}_{ip}$ = is the mass flow rate of product entering the system $= Fp_i$.

M_p = is the mass of the product in the reactor $= pV$.

$\hat{M}_{op}$ = is the mass flow rate of product output = 0.

$R_{Gp} = r_g V$

$R_{Cp} = 0$

$$\frac{dM}{dt} = \frac{d(pV)}{dt} = p_i F + r_g V \qquad (3.42)$$

$$V\frac{dp}{dt} = p_i F + r_g V - pF = r_g V + F(p_i - p) \qquad (3.43)$$

$$\frac{dp}{dt} = r_g + \frac{F}{V}(p_i - p) = Y_{p/x}\mu x + \frac{F}{V}(p_i - p) \qquad (3.44)$$

Normally $p_i = 0$ for sterile input flow F and then

$$\frac{dp}{dt} = Y_{p/x}\mu x + \frac{F}{V}p \tag{3.45}$$

3.4.5 Volume mass balance equation

In fed-batch process M_v is the volume V, where

$M_v = V$

$M_{iv} = F$

$\hat{M}_{ov} = 0$

$R_{GV} = 0$

$R_{CV} = 0$, then

$$\frac{dM}{dt} = \frac{dV(t)}{dt} = F(t), V(0) = Vo, V(t_f) = V_{max}$$

3.4.6 Summary of derived mass balance equations

The set of mass balance equations in function of time determines the process behaviour. Then the model of the process is:

$$\dot{x}(t) = x(t)(\mu(t) - k_d - D(t)), x(0) = x_0 \tag{3.46}$$

$$\dot{s}(t) = D(t)(s_i(t) - s(t)) - x(t)(\frac{2\mu(t)}{Y_{x/s}} + m_s), s(0) = s_0 \tag{3.47}$$

$$\dot{p}(t) = Y_{p/x}\mu(t)x(t) + Dp(k) \tag{3.48}$$

$$\dot{V}(t) = F(t), V(0) = V_0 \tag{3.49}$$

$$D(t) = \frac{F(t)}{V(t)} \tag{3.50}$$

$$\mu(t) = \frac{\mu_m(t)s(t)}{K_s + s(t)} \tag{3.51}$$

For inhibition of the cell growth by the ethanol production, the Monod equation can be written as:

$$\mu(t) = (1 - \frac{p(t)}{p^*}) \frac{\mu_m(t)s(t)}{K_s + s(t)} \quad (3.52)$$

Once the values of μ_{max}, K_s, $Y_{x/s}$ and $Y_{p/s}$ have been determined, the model is used to estimate cell biomass, substrate and product concentration as a function of time. The main problem with the fermentation process is that model parameters are difficult to be calculated because they change with time.

3.5 Incorporation of physiochemical variables into the biological model

The influence of physiochemical variables i.e. T, pH, DO_2 is very important for the process behaviour. These variables influence the cell enzymes and through them influence the biomass growth or product formation. The complex relationship between physiochemical and biological variables and sudden unexplained changes in the process is reflected in the values of the equation kinetic parameters μ_{max}, K_s, $Y_{x/s}$ and $Y_{p/x}$. They can be represented as functions of the physiochemical variables in the following way:

$$\mu_{max} = a_1 + a_2T + a_3T^2 + a_4pH + a_5pH^2 + a_6DO_2 + a_7D0^2 \quad (3.53)$$

$$k_s = b_1 + b_2T + b_3T^2 + b_4pH + b_5pH^2 + b_6DO_2 + b_7D0^2 \quad (3.54)$$

$$Y_{x/s} = c_1 + c_2T + c_3T^2 + c_4pH + c_5pH^2 + c_6DO_2 + c_7D0^2 \quad (3.55)$$

$$Y_{p/x} = d_1 + d_2T + d_3T^2 + d_4pH + d_5pH^2 + d_6DO_2 + d_7D0^2 \quad (3.56)$$

The mass balance equations (3.46)-(3.49) together with equations (3.53)-(3.56) represent the model that is used in the dissertation for process optimisation and control. Input variables for this model are the physiochemical variables T, pH, DO2, output variables are the biological variables x, s, p, where p is the product of ethanol giving inhibitory effect on the cell growth.

The coefficients values of $a_1 \rightarrow a_7$, $b_1 \rightarrow b_7$, $c_1 \rightarrow c_7$, and $d_1 \rightarrow d_7$ are not known and also they cannot be determined theoretically. As a result of the latter, these coefficients are

determined on the basis of experimental data and application of the least mean square [LSM] approach for the parameter estimation.

In equations (3.53)-(3.56) the value of the coefficients k_d and m are determined previously. Usually these values are very small approximately k_d = 0.01hr-1, m = 0.03 [g of substrate/g.cells.hr].

Model (3.53)-(3.56) is a nonlinear one according to both biological variables and coefficients.

3.6 Discrete representation of the model

In order to do calculations easily in the PC, model equations are discretized using difference equations for representation of the derivatives. The model is then described by difference equations.

Representation of the derivative:

$$\dot{x}(t) = \frac{x(k+1) - x(k)}{\Delta t}, \tag{3.57}$$

where Δt is a sampling period. The number of steps in the period of fermentation is $K = t_f/\Delta t$, where t_f is the end time of the fermentation.

Discrete model is obtained after substitution equation (3.57) into the continuous model equations (3.46)-(3.49).

Discrete model:

$$\frac{x(k+1) - x(k)}{\Delta t} = \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)}{K_s + s(k)}\right] - k_d x(k) - \frac{F(k)x(k)}{V(k)}, x(0) = x_0 \tag{3.58}$$

$$\frac{s(k+1) - s(k)}{\Delta t} = -\frac{\mu_{\max}}{Y_{x/s}}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)}{K_s + s(k)}\right] - mx(k) + \frac{F(k)}{V(k)}[s_i - s(k)], s(0) = s_0 \tag{3.59}$$

$$\frac{p(k+1)-p(k)}{\Delta t}=\mu_{\max}Y_{p/x}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{K_s+s(k)}\right]-\Delta t\frac{F(k)p(k)}{V(k)},p(0)=0 \tag{3.60}$$

$$\frac{V(k+1)-V(k)}{\Delta t}=F(k),V(0)=V_0,V(k)=V_{\max} \tag{3.61}$$

The above equations can be written in the following way:

$$x(k+1)=x(k)+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{K_s+s(k)}\right]-\Delta tk_d x(k)-\Delta t\frac{F(k)x(k)}{V(k)},x(0)=x_0 \tag{3.62}$$

$$s(k+1)=s(k)-\frac{\Delta t\mu_{\max}}{Y_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{K_s+s(k)}\right]-\Delta tmx(k)+\Delta t\frac{F(k)}{V(k)}[s_i-s(k)],s(0)=s_0 \tag{3.63}$$

$$p(k+1)=p(k)+\Delta t\mu_{\max}Y_{p/x}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{K_s+s(k)}\right]-\Delta t\frac{F(k)p(k)}{V(k)},p(0)=0 \tag{3.64}$$

$$V(k+1)=V(k)+\Delta tF(k),V(0)=V_0,V(k)=V_{\max} \tag{3.65}$$

$$D(k)=\frac{F(k)}{V(k)} \tag{3.66}$$

where the kinetic parameters $\mu_{\max},K_s,Y_{x/s},Y_{p/s}$ are described by the equations (3.53)-(3.56). The parameters $a_1 \rightarrow a_7$, $b_1 \rightarrow b_7$, $c_1 \rightarrow c_7$, and $d_1 \rightarrow d_7$ are unknown. They are determined on the basis of experimental data and estimation method in the next chapter.

3.7 Simulation of the model

Simulations of the model is realized using the packages:

→ MATLAB 6.1 / Simulink

These packages are used by scientist, engineers, researchers, industry etc for calculation, simulation and real time control. Matlab is an acronym for MATrix. Simulink is an extension of Matlab that allows development of models using block diagrams notation.
To allow easy of testing the model is developed in MATLAB. The input parameters and constants are the same to both models. Table 3.1 and Table 3.2 have the input and output variables for both simulation packages and they are used in the next paragraphs.

Variable	Value	Description
TT	50	→ Number of samples
t	1:1:TT	→ Time [h]
F	0	→ Volumetric substrate flow rate [mg/l].
si	0	→ Input substrate [mg/l]
pH	6.7	→ pH
T	30	→ Temperature in degrees Celsius
O	1	→ Dissolved Oxygen (DO_2) is %.
a1	0.08	→ Model parameter.
a2	0.03	→ Model parameter.
a3	0.01	→ Model parameter.
a4	0.02	→ Model parameter.
a5	0.002	→ Model parameter.
a6	0.001	→ Model parameter.
a7	0.002	→ Model parameter.
b1	0.2	→ Model parameter.
b2	0.03	→ Model parameter.
b3	0.01	→ Model parameter.
b4	0.01	→ Model parameter.
b5	0.001	→ Model parameter.
b6	0.002	→ Model parameter.
b7	0.001	→ Model parameter.
c1	2	→ Model parameter.

c2	0.3	→ Model parameter.
c3	0.021	→ Model parameter.
c4	0.051	→ Model parameter.
c5	0.00211	→ Model parameter.
c6	0.0211	→ Model parameter.
c7	0.002	→ Model parameter.
d1	0.004	→ Model parameter.
d2	0.006	→ Model parameter.
d3	0.001	→ Model parameter.
d4	0.005	→ Model parameter.
d5	0.0005	→ Model parameter.
d6	0.001	→ Model parameter.
d7	0.0002	→ Model parameter.
k_d	0.09	→ Death co-efficient k_d.
pl	50	→ Inhibition factor p*.
m	0.0011	→ Maintenance co-efficient.
n	0.52	→ power coefficient

Table 3.1 Input variables to the yeast model

Variable	**Value**	**Description**
X		Biomass concentration [mg/l]
S		Substrate concentration [mg/l]
P		Product concentration [mg/l]
u_{max}		Maximum growth rate
K_s		Substrate saturation / Monod's constant
$Y_{x/s}$		Stochiometry coefficient for biomass growth
$Y_{x/p}$		Stochiometry coefficient for product growth

Table 3.2 Output variables from the yeast model

3.8 Results from simulation

The results from simulation will be described in chapter 6.

3.9 Conclusion

Many aspects of fermentation are poorly understood, the latter makes it difficult to derive a model that will cover all the areas of interest. Evidence that all important fermentation variables have not yet been identified is the significant batch-to-batch variation that occurs in the fermentation industry.

The key fermentation variables cannot be measured on-line. Delays are often experienced during the off-line measurements of the concentrations of substrate, biomass and the product.

The developed model is semi-theoretical in the sense that mass balance equations are theoretical mathematical model, but coefficient cannot be calculated mathematically. These coefficients are introduced mathematically to represent the influence of the physiochemical variables to the biological variables. The latter is achieved by representing the kinetic parameters of biological model through quadratic functions of physiological variables.

Parameters estimation is used to determine the coefficients of these functions.

The utilization of Matlab/Simulink packages made the development much easier when it comes to testing of the prototype. The results received on both packages were
the same, proving the operation of the algorithm and also the consistence was tested.

The results from simulation are used later for model parameter estimation and process optimization.

Next chapter will be introduced that using optimal control theory to solve the optimal control problem, the approach of the functional of Lagrange.

Chapter 4

Optimal control of the fed-batch fermentation process for production of yeast

4. Introduction

The optimal control problem is solved using optimal control theory, the approach of the functional of Lagrange. The optimisation layer problems are determined and based on

- the determined from the previous layer model
- some criteria for the production of biomass, product
- some constraints over the minimal and maximal values of variables.

In the study it is based on maximizing production of yeast. Two types of problems are solved on the layer of optimisation:

Problem for optimal control of the process

Problem for tuning of the PID controller parameters for direct control implementation

The solution of the first problem determines the solution of the second one. In the considered case the problem for optimal control is formulated in the following way : Find the trajectories of the temperature T, pH, dissolved oxygen and the stirrer rpm in a such a way that the production of biomass at the end of fermentation is maximum. Formulation of the problem, description of the developed methods, algorithms and programs is given in the chapter.

4.1 Problem for optimal control of the production of yeast

The aim in producing yeast, needed, as inoculant in beer fermentation is maximum quantity of biomass, which means maximum concentration of biomass at the end of the fermentation process. The problem is how to influence the process in order to achieve this aim. The possibilities for influence are the input substrate flow rate, temperature, pH and DO_2 concentration, which in this study are considered as control variables. Their optimal

trajectories can be obtained through the formulation and solution of the problem for optimal control of the process.

4.1.1 Formulation of the problem for optimal control

Find the trajectories of the feed flow rate, temperature, pH and dissolved oxygen concentration: $F(k), T(k), pH(k), DO_2(k), k = \overline{0, K-1}$ in such a way that the concentration of biomass at the end of the process

$$J = x(K) \to \max \tag{4.1}$$

is maximized under the model equations:

$$x(k+1) = x(k) + \Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)}{K_s + s(k)}\right] - \Delta t k_d x(k) - \Delta t \frac{F(k)x(k)}{V(k)} = f_x(k),$$

$$x(0) = x_0 \tag{4.2}$$

$$s(k+1) = s(k) - \frac{\Delta t \mu_{\max}}{Y_{x/s}}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)}{K_s + s(k)}\right] - \Delta t m x(k) + \Delta t \frac{F(k)}{V(k)}[s_i - s(k)] = f_s(k),$$

$$s(0) = s_0 \tag{4.3}$$

$$p(k+1) = p(k) + \Delta t \mu_{\max} Y_{p/x}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)}{K_s + s(k)}\right] - \Delta t \frac{F(k)p(k)}{V(k)} = f_p(k), p(0) = 0 \tag{4.4}$$

$$V(k+1) = V(k) + \Delta t F(k) = f_v(k), \quad V(0) = V_0, \quad V(K) = V_{\max} \tag{4.5}$$

$$\mu_{\max} = a_1 + a_2 T(k) + a_3 T^2(k) + a_4 pH(k) + a_5 pH^2(k) + a_6 DO_2(k) + a_7 D0^2{}_2(k) \tag{4.6}$$

$$K_s = b_1 + b_2 T(k) + b_3 T^2(k) + b_4 pH(k) + b_5 pH^2(k) + b_6 DO_2(k) + b_7 D0^2{}_2(k) \tag{4.7}$$

$$Y_{x/s} = c_1 + c_2 T(k) + c_3 T^2(k) + c_4 pH(k) + c_5 pH^2(k) + c_6 DO_2(k) + c_7 D0^2{}_2(k) \tag{4.8}$$

$$Y_{p/x} = d_1 + d_2 T(k) + d_3 T^2(k) + d_4 pH(k) + d_5 pH^2(k) + d_6 DO_2(k) + d_7 D0^2{}_2(k) \tag{4.9}$$

and under limitations over the minimal and maximal values of the process variables

$$\begin{aligned}
&X_{\min}(k) \le X(k) \le X_{\max}(k) \\
&S_{\min}(k) \le S(k) \le S_{\max}(k) \\
&T_{\min}(k) \le T(k) \le T_{\max}(k) \\
&pH_{\min}(k) \le pH(k) \le pH_{\max}(k) \\
&DO_{\min}(k) \le DO(k) \le DO_{\max}(k) \\
&rpm_{\min}(k) \le rpm(k) \le rpm_{\max}(k)
\end{aligned} \tag{4.10}$$

where k is the discrete time, and K is the final moment.

4.1.2 Method for problem solution

The qualities of the process like an object of control lead to many difficulties in solving the optimal control problems such as non-linear two point boundary value problems have to be solved and singular kind of control to be calculated [SKOGESTAK *et al.*, 1999]. A decomposition method for solving the discrete time formulation of the problem is proposed in the thesis based on Variation calculus using an augmented LaGrange functional and its decomposition in time domain. Some partial developments of this method have been given for secondary metabolites and xanthan gum production. The method is developed for a general description of the batch fermentation by means of non-structured models with slowly varying parameters. The method is characterized by a new coordinating vector that permits an augmented LaGrange functional to be used for solving the non-convex optimal control problem, and for preserving the full decomposition in time domain of the dual problem. The decomposition method gives two possibilities to overcome some of the

mentioned drawbacks of other methods of singular control for optimal profile calculation [SING and TITLI., 1978]. The programmes for real time calculation will be carried out using LabVIEW. The simulations with the obtained control will be carried out using MATALAB and LabVIEW packages. The optimal profiles of the chemical variables are the end solution to the optimization problem. They are not constant and they present the desired outputs for the local PID control loops for the temperature, pH, dissolved oxygen, etc. These trajectories will be send to the layer of direct control as set-points for the controllers. The used augmented functional of Lagrange has the form:

$$L = x(k) + \sum_{k=0}^{k-1} \begin{Bmatrix} \lambda_x(k)[-x(k+1) + f_x(k)] + \lambda_s(k)[-s(k+1) + f_s(k)] + \\ \alpha_x[-x(k+1) + f_x(k)]^2 + \alpha_s[-s(k+1) + f_s(k)]^2 \end{Bmatrix} \tag{4.11}$$

where λ are the costate variables and α are the penalty coefficients.

4.1.2.1 Necessary conditions for optimality

The solution of the optimal control problem is based on the necessary conditions for the optimality of the functional of Lagrange [WILLIAMS D., YOUSEFPOUR P., WELLINGTON E., 1986], [TEO, K.L.; GOH, C.J. &WONG, K.H. 1991]. The necessary conditions for optimality are:

For the state variables:

Biomass

$$\frac{\partial L}{\partial x(k)} = 0, \quad \frac{\partial L}{\partial s(k)} = 0, \quad \frac{\partial L}{\partial p(k)} = 0, \quad \frac{\partial L}{\partial V(k)} = 0, \tag{4.12}$$

For the control variables

$$\frac{\partial L}{\partial T(k)} = 0, \quad \frac{\partial L}{\partial pH(k)} = 0, \quad \frac{\partial L}{\partial DO_2(k)} = 0, \quad \frac{\partial L}{\partial F(k)} = 0, \tag{4.13}$$

For the conjugate variables

$$\frac{\partial L}{\partial \lambda_x(k)} = 0, \quad \frac{\partial L}{\partial \lambda_s(k)} = 0, \quad \frac{\partial L}{\partial \lambda_p(k)} = 0, \quad \frac{\partial L}{\partial \lambda_V(k)} = 0. \tag{4.14}$$

The analytical expressions for the derivatives is represented in the following way:

For the state variables

$$\frac{\partial L}{\partial x(k)} = \lambda_x(k)\left[1+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s+s(k)}\right]-\Delta tk_d-\frac{\Delta tF(k)}{V(k)}\right]+$$

$$\mu_x(k)[x(k+1)-f_x(k)]\left[1+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s+s(k)}\right]-\Delta tk_d-\frac{\Delta tF(k)}{V(k)}\right]+$$

$$\lambda_s(k)\left[\frac{\Delta t\mu_{\max}}{Y_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s+s(k)}\right]-\Delta tm\right]-$$

$$-\mu_s(k)[s(k+1)-f_s(k)]\left[\frac{\Delta t\mu_{\max}}{Y_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{Ks+s(k)}\right]-\Delta tm\right]+$$

$$+\lambda_p(k)\left[\Delta tY_{p/x}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s+s(k)}\right]\right]-$$

$$-\mu_p(k)[p(k+1)-f_p(k)]\left[\Delta tY_{p/x}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{s(k)}{K_s+s(k)}\right]\right]=0=e_x(k)$$

$$\frac{\partial L}{\partial x(K)}=1+\lambda_x(K)+\mu_x(K) \qquad (4.15)$$

Substrate

$$\frac{\partial L}{\partial s(k)} = \lambda_x(k)\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)(K_s+s(k))-x(k)s(k)}{[K_s+s(k)]^2}\right]\right]-$$

$$-\mu_x(k)[x(k+1)-f_x(k)]\left[\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{[K_s+s(k)]^2}\right]\right]+$$

$$+\lambda_s(k)\left[1-\frac{\Delta t\mu_{\max}}{Y_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{[K_s+s(k)]^2}\right]-\frac{\Delta tF(k)}{V(k)}\right]-$$

$$-\mu_s(k)[s(k+1)-f_s(k)]\left[1-\frac{\Delta t\mu_{\max}}{Y_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{[K_s+s(k)]^2}\right]-\frac{\Delta tF(k)}{V(k)}\right]$$

$$+\lambda_p(k)\left[\Delta t Y_{p/x}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{[K_s+s(k)]^2}\right]\right]-$$

$$-\mu_p(k)[p(k+1)-f_p(k)]\left[\Delta t Y_{p/x}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)}{[K_s+s(k)]^2}\right]\right]=0=e_s(k),k=\overline{0,K-1}$$

$$\frac{\partial L}{\partial s(K)}=\lambda_s(K)+\mu_s(K) \tag{4.16}$$

Product

$$\frac{\partial L}{\partial p(k)}=\lambda_p(k)\left[1+\Delta t Y_{p/x}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1}\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]-\frac{\Delta t F(k)}{V(k)}\right]-$$

$$-\mu_p(k)[p(k+1)-f_p(k)]\left[1+\Delta t Y_{p/x}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^{n-1}\left[-\frac{1}{p^*}\right]\left[\frac{x(k)s(k)}{K_s+s(k)}\right]-\frac{\Delta t F(k)}{V(k)}\right]$$

$$=0=e_p(k),\ k=\overline{0,K-1}$$

$$\frac{\partial L}{\partial p(K)}=\lambda_p(K)+\mu_p(K) \tag{4.17}$$

Volume

$$\frac{\partial L}{\partial V(k)}=\lambda_x(k)\left[-\Delta t\frac{-F(k)x(k)}{V^2(k)}\right]-\mu_x(k)[x(k+1)-f_x(k)]\left[\Delta t\frac{F(k)x(k)}{V^2(k)}\right]+$$

$$+\lambda_s(k)\left[\Delta t\frac{-F(k)}{V^2(k)}[s_i-s(k)]\right]-\mu_s(k)[s(k+1)-f_s(k)]\left[\Delta t\frac{-F(k)}{V^2(k)}[s_i-s(k)]\right]+$$

$$+\lambda_p(k)\left[-\Delta t\frac{-F(k)x(k)}{V^2(k)}\right]-\mu_s(k)[p(k+1)-f_p(k)]\left[\Delta t\frac{F(k)p(k)}{V^2(k)}\right]+$$

$$+\lambda_v(k)[1]-\mu_v(k)[V(k)+1)-f_v(k)].[1]=0=e_v(k),k=\overline{0,K-1}$$

$$\frac{\partial L}{\partial V(K)}=\lambda_V(K)+\mu_V(K) \tag{4.18}$$

For the conjugate variables

$$\frac{\partial L}{\partial \lambda_x(k)}=x(k+1)-f_x(k)=0=e_{\lambda x}(k) \tag{4.19}$$

$$\frac{\partial L}{\partial \lambda_s(k)} = s(k+1) - f_s(k) = 0 = e_{\lambda s}(k) \tag{4.20}$$

$$\frac{\partial L}{\partial \lambda_p(k)} = p(k+1) - f_p(k) = 0 = e_{\lambda p}(k) \tag{4.21}$$

$$\frac{\partial L}{\partial \lambda_v(k)} = V(k+1) - f_v(k) = 0 = e_{\lambda v}(k) \tag{4.22}$$

. For the control variables

Flow rate

$$\frac{\partial L}{\partial F(k)} = \lambda_x(k)\left[\frac{-x(k)\Delta t}{V(k)}\right] - \mu_x(k)[x(k+1) + f_x(k)]\left[\frac{\Delta t x(k)}{V(k)}\right] +$$
$$+\lambda_s(k)\left[\frac{\Delta t[s_i - s(k)]}{V(k)}\right] - \mu_s(k)[s(k+1) - f_s(k)]\left[\frac{\Delta t[s_i - s(k)]}{V(k)}\right] +$$
$$+\lambda_p(k)\left[\frac{-\Delta t p(k)}{V(k)}\right] - \mu_p(k)[p(k+1) - f_p(k)]\left[\frac{-\Delta t p(k)}{V(k)}\right] +$$
$$+\lambda_v(k)\Delta t - \mu_v(k)[V(k)+1) - f_v(k)]\Delta t = 0 = e_F(k),\ k = \overline{0, K-1} \tag{4.23}$$

Temperature

$$\frac{\partial L}{\partial T(k)} = \lambda_x(k)\left[\frac{\Delta t \partial \mu_{\max}}{\partial T}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s + s(k)} + \Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s + s(k)]^2}\right]\right] -$$
$$-\mu_x(k)[x(k+1) -$$
$$-f_x(k)]\left[\frac{\Delta t \partial \mu_{\max}}{\partial T}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s + s(k)} + \Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s + s(k)]^2}\right]\right] +$$
$$+\lambda_s(k)\left[-\Delta t \frac{\frac{\partial \mu_{\max}}{\partial T} Y_{x/s} - \mu_{\max}\frac{\partial Y_{x/s}}{\partial T}}{Y^2{}_{x/s}}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s + s(k)} -\right.$$

$$-\frac{\Delta t\mu_{\max}}{Y_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\dfrac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]-$$

$$-\mu_s(k)[s(k+1)-f_s(k)]\left[-\Delta t\frac{\dfrac{\partial\mu_{\max}}{\partial T}Y_{x/s}-\mu_{\max}\dfrac{\partial Y_{x/s}}{\partial T}}{Y^2{}_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}-\right.$$

$$\left.-\frac{\Delta t\mu_{\max}}{Y_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\dfrac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\right]+$$

$$+\lambda_p(k)\left[\Delta t\left[\frac{\partial Y_{p/x}}{\partial T}\mu_{\max}+Y_{p/x}\frac{\partial\mu_{\max}}{\partial T}\right]\frac{x(k)s(k)}{K_s+s(k)}+\right.$$

$$+\Delta tY_{p/x}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\dfrac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]-$$

$$-\mu_p(k)[p(k+1)-f_p(k)]\left[\Delta t\left[\frac{\partial Y_{p/x}}{\partial T}\mu_{\max}+Y_{p/x}\frac{\partial\mu_{\max}}{\partial T}\right]\frac{x(k)s(k)}{K_s+s(k)}+\right. \qquad (4.24)$$

$$\left.+\Delta tY_{p/x}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\dfrac{\partial K_s}{\partial T}}{[K_s+s(k)]^2}\right]\right]=0=e_T(k)$$

pH

$$\frac{\partial L}{\partial pH(k)}=\lambda_x(k)\left[\Delta t\frac{\partial\mu_{\max}}{\partial pH}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\dfrac{\partial K_s}{\partial pH}}{[K_s+s(k)]^2}\right]\right]$$

$$-\mu_x(k)[x(k+1)-f_x(k)]\left[\Delta t\frac{\partial\mu_{max}}{\partial pH}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\dfrac{\partial K_s}{\partial pH}}{[K_s+s(k)]^2}\right]\right]$$

$$+\lambda_s(k)\left[-\Delta t\frac{\dfrac{\partial\mu_{\max}}{\partial pH}Y_{x/s}-\mu_{\max}\dfrac{\partial Y_{x/s}}{\partial pH}}{Y^2{}_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}-\right.$$

$$-\frac{\Delta t\mu_{\max}}{Y_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\dfrac{\partial K_s}{\partial pH}}{[K_s+s(k)]^2}\right]-$$

$$-\mu_s(k)[s(k+1)-f_s(k)]+\left[-\Delta t\frac{\dfrac{\partial\mu_{\max}}{\partial pH}Y_{x/s}-\mu_{\max}\dfrac{\partial Y_{x/s}}{\partial pH}}{Y^2{}_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}-\right.$$

$$-\frac{\Delta t\mu_{\max}}{Y_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\dfrac{\partial K_s}{\partial pH}}{[K_s+s(k)]^2}\right]+$$

$$+\lambda_p(k)\left[\Delta t\left[\frac{\partial Y_{p/x}}{\partial pH}\mu_{\max}+Y_{p/x}\frac{\partial\mu_{\max}}{\partial pH}\right]\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\right.$$

$$+\Delta tY_{p/x}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\dfrac{\partial K_s}{\partial pH}}{[K_s+s(k)]^2}\right]\Bigg]-$$

$$-\mu_p(k)\left[\Delta t\left[\frac{\partial Y_{p/x}}{\partial pH}\mu_{\max}+Y_{p/x}\frac{\partial\mu_{\max}}{\partial pH}\right]\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}\right. \qquad (4.25)$$

$$\left.+\Delta tY_{p/x}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\dfrac{\partial K_s}{\partial pH}}{[K_s+s(k)]^2}\right]\right]=0=e_{ph}(k)$$

Dissolved oxygen

$$\frac{\partial L}{\partial DO_2(k)}=\lambda_x(k)\left[\Delta t\frac{\partial\mu_{\max}}{\partial DO_2}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\dfrac{\partial K_s}{\partial DO_2}}{[K_s+s(k)]^2}\right]\right]$$

$$-\mu_x(k)[x(k+1)-f_x(k)]$$

$$\left[\Delta t\frac{\partial\mu_{max}}{\partial DO_2}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\dfrac{\partial K_s}{\partial DO_2}}{[K_s+s(k)]^2}\right]\right]$$

$$+\lambda_s(k)\left[-\Delta t\frac{\frac{\partial\mu_{\max}}{\partial DO_2}Y_{x/s}-\mu_{\max}\frac{\partial Y_{x/s}}{\partial DO_2}}{Y^2{}_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}\right.$$

$$\left.-\frac{\Delta t\mu_{\max}}{Y_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial DO_2}}{[K_s+s(k)]^2}\right]-\right.$$

$$\mu_s(k)[s(k+1)-f_s(k)]+\left[-\Delta t\frac{\frac{\partial\mu_{\max}}{\partial DO_2}Y_{x/s}-\mu_{\max}\frac{\partial Y_{x/s}}{\partial DO_2}}{Y^2{}_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}-\right.$$

$$\left.-\frac{\Delta t\mu_{\max}}{Y_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial DO_2}}{[K_s+s(k)]^2}\right]+\right.$$

$$+\lambda_p(k)\left[\Delta t\left[\frac{\partial Y_{p/x}}{\partial DO_2}\mu_{\max}+Y_{p/x}\frac{\partial\mu_{\max}}{\partial DO_2}\right]\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\right.$$

$$\left.+\Delta tY_{p/x}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial DO_2}}{[K_s+s(k)]^2}\right]\right]-$$

$$\mu_p(k)[p(k+1)-f_p(k)]\left[\Delta t\left[\frac{\partial Y_{p/x}}{\partial DO_2}\mu_{\max}+Y_{p/x}\frac{\partial\mu_{\max}}{\partial DO_2}\right]\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}\right.$$

$$\left.+\Delta tY_{p/x}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial DO_2}}{[K_s+s(k)]^2}\right]\right]=0=e_{DO_2}(k),\quad k=\overline{0,K-1},$$

(4.26)

where the partial derivatives of the kinetic parameters are:

According to the temperature

$$\left.\begin{aligned} &\frac{\partial \mu_{\max}}{\partial T} = a_2 + 2a_3 T(k), \frac{\partial K_s}{\partial T} = b_2 + 2b_3 T(k), \\ &\frac{\partial Y_{x/s}}{\partial T} = c_2 + 2c_3 T(k), \frac{\partial Y_{p/s}}{\partial T} = d_2 + 2d_3 T(k), \end{aligned}\right\} \quad (4.27)$$

According to the DO_2

$$\left.\begin{aligned} &\frac{\partial \mu_{\max}}{\partial DO_2} = a_6 + 2a_7 DO_2, \frac{\partial K_s}{\partial DO_2} = b_6 + 2b_7 DO_2(k), \\ &\frac{\partial Y_{x/s}}{\partial DO_2} = c_6 + 2c_7 DO_2(k), \frac{\partial Y_{p/x}}{\partial DO_2} = d_6 + 2d_7 DO_2(k), \end{aligned}\right\} \quad (4.28)$$

According to the pH

$$\left.\begin{aligned} &\frac{\partial \mu_{\max}}{\partial pH} = a_4 + 2a_5 pH(k), \frac{\partial K_s}{\partial pH} = b_4 + 2b_5 pH(k), \\ &\frac{\partial Y_{x/s}}{\partial pH} = c_4 + 2c_5 pH(k), \frac{\partial Y_{p/x}}{\partial pH} = d_4 + 2d_5 pH(k), \end{aligned}\right\} \quad (4.29)$$

$k = \overline{0, K-1}$

The necessary conditions for optimality form a set of interconnected nonlinear equations , which is necessary to be solved. Because the solution is difficult to handle analytically, decomposition in the time domain of the necessary conditions for optimality will be introduced.

4.1.2.2 Coordinating vector

The overall control problem is decomposed into one coordinating and k+1 sub-problems. The process of solution is iterative one. The coordinator selects the values of the coordinating variables when they are substituted into the functional of Lagrange, its full decomposition in the time domain is obtained. The sub-problems on the first level are solved with the fixed values of the coordinating variables. Their solutions are found for the separate time moments of the optimization horizon. Then these solutions are sent to the coordinator, where an improved value of the coordinating variables is calculation conditions for termination of calculation is checked.

If it is fulfilled, the optimal values of the coordinating variables are obtained. These values determine the optimal values of the sub-problems solutions, because of the connection of all solutions through the necessary conditions for optimality. If the conditions for termination are not fulfilled, the iterative are repeated, and so on.

The above algorithm will be implemented first in Matlab software environment in two ways:

— sequentially in one computer

— parallel, using the cluster of computers

The decomposition in time domain is based on a two level calculation structure, given in Fig 4.1

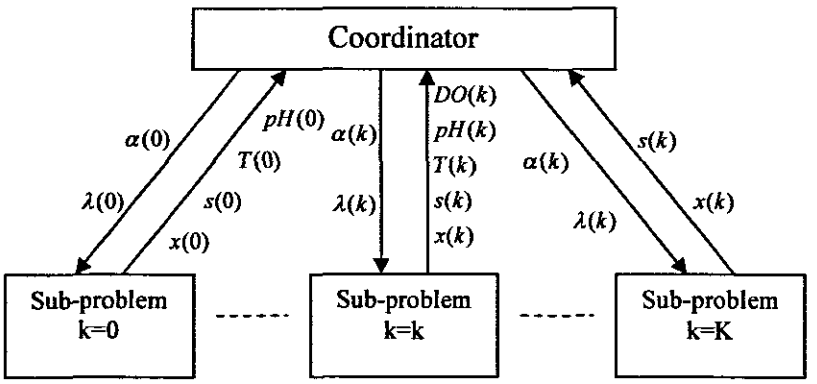


Fig 4.1 Two level calculation structure

4.1.2.3 First level sub-problems

The solutions of the first level sub-problems represent solution of the set of equations (4.16) – (4.19), (4.23) – (4.29) in which the coordinating variables are substituted. These equations are solved for every separate moment of time in a parallel way. It can be seen from the structure of the equations that analytical solution is impossible. The gradient procedure can be used in the following way

$$v^{j,i+1}(k) = v^{j,i}(k) + \alpha_v e_v{}^{j,i}(k), v = x, s, p, v, F, T, pH, DO_2 \qquad (4.30)$$

where $\alpha_v > 0$ are the steps of the gradient procedures, $e_v(k)$ are the gradients according to the equations (5.21)-(5.35) and i is the iteration index of the procedures. The equations (5.21)-(5.35) can be simplified in the following way:

For the state variables

$$\frac{\partial L}{\partial x(k)} = e_x^{j,i}(k) = \left[1 + \Delta t\mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{s(k)}{K_s + s(k)} - \Delta t k_d - \Delta t\frac{F(k)}{V(k)}\right] +$$
$$+\left[\frac{\Delta t\mu_{\max}}{Y_{p/s}}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{s(k)}{K_s + s(k)} - \Delta t m\right]\left[\lambda_x^{\,j}(k) - \mu_x e_{\lambda x}^{\,j}(k)\right] +$$
$$+\left[\Delta t Y_{p/x}\mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{s(k)}{K_s + s(k)}\right]\left[\lambda_p^{\,j}(k) - \mu_p e_{\lambda p}^{\,j}(k)\right] \tag{4.31}$$

$$\frac{\partial L}{\partial s(k)} = e_s^{j,i}(k) = \left[\Delta t\mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{\left[K_s + s(k)\right]^2}\right]\left[\lambda_x(k) - \mu_x e_{\lambda x}(k)\right] +$$
$$+\left[1 - \frac{\Delta t\mu_{\max}}{Y_{x/s}}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{s(k)x(k)}{\left[K_s + s(k)\right]^2} - \Delta t\frac{F(k)}{V(k)}\right]\left[\lambda_s^{\,j}(k) - \mu_s e_{\lambda s}^{\,j}(k)\right] +$$
$$+\left[\Delta t Y_{p/x}\mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s + s(k)}\right]\left[\lambda_p^{\,j}(k) - \mu_p e_{\lambda p}^{\,j}(k)\right] \tag{4.32}$$

$$\frac{\partial L}{\partial p(k)} = e_p^{j,i}(k) =$$
$$= \left[1 + \Delta t Y_{p/x}\mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^{n-1}\left[\frac{-1}{p^*}\right]\frac{x(k)s(k)}{K_s + s(k)} - \Delta t\frac{F(k)}{V(k)}\right]\left[\lambda_p^{\,j}(k) - \mu_p e_{\lambda p}^{\,j}(k)\right] \tag{4.33}$$

$$\frac{\partial L}{\partial V(k)} = e_V^{j,i}(k) = \left[\Delta t\frac{F(k)x(k)}{V^2(k)}\right]\left[\lambda_x(k) - \mu_x e_{\lambda x}(k)\right] +$$
$$+\left[-\frac{\Delta t F(k)}{V^2(k)}\left[s_i - s(k)\right]\right]\left[\lambda_s^{\,j}(k) - \mu_s e_{\lambda s}^{\,j}(k)\right] +$$
$$+\left[\Delta t\frac{F(k)p(k)}{V^2(k)}\right]\left[\lambda_p^{\,j}(k) - \mu_p e_{\lambda p}^{\,j}(k)\right] +$$
$$+\left[\lambda_V^{\,j}(k) - \mu_V e_{\lambda V}^{\,j}(k)\right] \tag{4.34}$$

For the control variables

$$\frac{\partial L}{\partial F(k)} = e_F{}^{j,j}(k) = \left[\Delta t \frac{x(k)}{V(k)}\right]\left[\lambda_x(k) - \mu_x e_{\lambda x}(k)\right] +$$
$$+\left[-\frac{\Delta t}{V(k)}\left[s_i - s(k)\right]\right]\left[\lambda_s{}^j(k) - \mu_s e_{\lambda_s}{}^j(k)\right] +$$
$$+\left[-\Delta t \frac{p(k)}{V(k)}\right]\left[\lambda_p{}^j(k) - \mu_p e_{\lambda p}{}^j(k)\right] +$$
$$+\Delta t\left[\lambda_V{}^j(k) - \mu_V e_{\lambda V}{}^j(k)\right] = e_F(k) \qquad (4.35)$$

$$\frac{\partial L}{\partial pH(k)} = e_{pH}{}^{j,j}(k) =$$
$$= \left[\Delta t \frac{\partial \mu_{\max}}{\partial pH}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s + s(k)} - \Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)\frac{\partial K_s}{\partial pH}}{\left[K_s + s(k)\right]^2}\right]\right] *$$
$$*\left[\lambda_x(k) - \mu_x(k) e_{\lambda x}(k)\right] +$$
$$+\left[-\Delta t \frac{\frac{\partial \mu_{\max}}{\partial pH} Y_{x/s} - \mu_{\max}\frac{\partial Y_{x/s}}{\partial pH}}{Y^2{}_{x/s}}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s + s(k)} + \right.$$
$$\left. + \frac{\Delta t \mu_{\max}}{Y_{x/s}}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)\frac{\partial K_s}{\partial pH}}{\left[K_s + s(k)\right]^2}\right]\right]\left[\lambda_s(k) - \mu_s(k) e_{\lambda s}(k)\right] +$$
$$+\left[\Delta t\left[\frac{\partial Y_{p/x}}{\partial pH}\mu_{\max} + Y_{p/x}\frac{\partial \mu_{\max}}{\partial pH}\right]\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s + s(k)} - \right.$$
$$\left. - \Delta t Y_{p/x}\mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^n \left[\frac{x(k)s(k)\frac{\partial K_s}{\partial pH}}{\left[K_s + s(k)\right]^2}\right]\right]\left[\lambda_p(k) - \mu_p(k) e_{\lambda p}(k)\right] \qquad (4.36)$$

$$\frac{\partial L}{\partial T(k)} = e_T(k) =$$

$$= \left[\frac{\Delta t \partial \mu_{\max}}{\partial T} \left[1 - \frac{p(k)}{p^*} \right]^n \frac{x(k)s(k)}{K_s + s(k)} - \Delta t \mu_{\max} \left[1 - \frac{p(k)}{p^*} \right]^n \left[\frac{-x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s + s(k)]^2} \right] \right] *$$

$$*\left[\lambda_x(k) - \mu_x(k) e_{\lambda x}(k) \right] +$$

$$+ \left[-\Delta t \frac{\frac{\partial \mu_{\max}}{\partial T} Y_{x/s} - \mu_{\max} \frac{\partial Y_{x/s}}{\partial T}}{Y^2{}_{x/s}} \left[1 - \frac{p(k)}{p^*} \right]^n \frac{x(k)s(k)}{K_s + s(k)} + \right.$$

$$\left. + \frac{\Delta t \mu_{\max}}{Y_{x/s}} \left[1 - \frac{p(k)}{p^*} \right]^n \left[\frac{x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s + s(k)]^2} \right] \right] \left[\lambda_s(k) - \mu_s(k) e_{\lambda s}(k) \right] +$$

$$+ \left[\Delta t \frac{\frac{\partial \mu_{\max}}{\partial T} Y_{x/s} + \mu_{\max} \frac{\partial Y_{x/s}}{\partial T}}{Y^2{}_{x/s}} \left[1 - \frac{p(k)}{p^*} \right]^n \frac{x(k)s(k)}{K_s + s(k)} - \right.$$

$$\left. - \Delta t \mu_{\max} Y_{p/x} \left[1 - \frac{p(k)}{p^*} \right]^n \left[\frac{x(k)s(k)\frac{\partial K_s}{\partial T}}{[K_s + s(k)]^2} \right] \right] \left[\lambda_p(k) - \mu_p(k) e_{\lambda p}(k) \right] \qquad (4.37)$$

$$\frac{\partial L}{\partial DO_2(k)} = e_{DO_2}{}^{jj}(k) =$$

$$= \left[\Delta t \frac{\partial \mu_{\max}}{\partial DO_2} \left[1 - \frac{p(k)}{p^*} \right]^n \frac{x(k)s(k)}{K_s + s(k)} - \Delta t \mu_{\max} \left[1 - \frac{p(k)}{p^*} \right]^n \left[\frac{x(k)s(k)\frac{\partial K_s}{\partial DO_2}}{[K_s + s(k)]^2} \right] \right] *$$

$$*\left[\lambda_x(k) - \mu_x(k) e_{\lambda x}(k) \right] +$$

$$+ \left[-\Delta t \frac{\frac{\partial \mu_{\max}}{\partial DO_2} Y_{x/s} - \mu_{\max} \frac{\partial Y_{x/s}}{\partial DO_2}}{Y^2{}_{x/s}} \left[1 - \frac{p(k)}{p^*} \right]^n \frac{x(k)s(k)}{K_s + s(k)} + \right.$$

$$+\frac{\Delta t\mu_{\max}}{Y_{x/s}}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)\dfrac{\partial K_s}{\partial DO_2}}{[K_s+s(k)]^2}\right]\left[\lambda_s(k)-\mu_s(k)e_{\lambda_s}(k)\right]+$$

$$+\left[\Delta t\left[\frac{\partial Y_{p/x}}{\partial DO_2}\mu_{\max}+Y_{p/x}\frac{\partial \mu_{\max}}{\partial DO_2}\right]\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}+\right.$$

$$\left.+\Delta tY_{p/x}\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\left[\frac{x(k)s(k)\dfrac{\partial K_s}{\partial DO_2}}{[K_s+s(k)]^2}\right]\right]\left[\lambda_p(k)-\mu_p(k)e_{\lambda_p}(k)\right]$$

(4.38)

These equations can be simplified further for the purpose of calculations. Some expressions in them are repeated, that is why the common notations are used to represent them:

$$\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\frac{s(k)}{K_s+s(k)}=\varphi_x(k) \tag{4.39}$$

$$\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{[K_s+s(k)]^2}=\varphi_s(k) \tag{4.40}$$

$$\Delta t\frac{F(k)}{V^2(k)}=\varphi_v(k) \tag{4.41}$$

$$\frac{\Delta t\mu_{\max}Y_{p/x}}{p^*}\left[1-\frac{p(k)}{p^*}\right]^{n-1}\frac{x(k)s(k)}{K_s+s(k)}=\varphi_p(k) \tag{4.42}$$

$$\Delta t\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{K_s+s(k)}=\varphi_1(k) \tag{4.43}$$

$$\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n\frac{x(k)s(k)}{[K_s+s(k)]^2}=\varphi_2(k),k=\overline{0,K-1} \tag{4.44}$$

Using the notations (4.39)-(4.44), the equations (4.31)-(4.38) can be written in the following way:

For state variables

$$\frac{\partial L}{\partial x(k)} = e_x{}^{j,J}(k) = \left[1 + \varphi_x{}^{j,J}(k) - \Delta t k_d + \Delta t \frac{F(k)}{V(k)}\right]\left[\lambda_x{}^{j}(k) - \mu_x(k) e_{\lambda x}{}^{j}(k)\right] +$$
$$+ \left[\frac{e_x{}^{j,J}(k)}{Y_{x/s}} - \Delta t m\right]\left[\lambda_s{}^{j}(k) - \mu_s(k) e_{\lambda s}{}^{j}(k)\right] +$$
$$+ \Delta t \varphi_x{}^{j,J}(k) Y_{p/x}\left[\lambda_p{}^{j}(k) - \mu_p(k) e_{\lambda p}{}^{j}(k)\right]$$

(4.45)

$$\frac{\partial L}{\partial s(k)} = e_s{}^{j,J}(k) = \varphi_s{}^{j,J}(k)\left[\lambda_x{}^{j}(k) - \mu_x(k) e_{\lambda x}{}^{j}(k)\right] +$$
$$+ \left[1 - \frac{\varphi_s{}^{j,J}(k)}{Y_{x/s}} - \Delta t \frac{F^{j,J}(k)}{V^{j,J}(k)}\right]\left[\lambda_s{}^{j}(k) - \mu_s(k) e_{\lambda s}{}^{j}(k)\right] +$$
$$+ Y_{p/x}\varphi_s(k)\left[\lambda_p{}^{j}(k) - \mu_p(k) e_{\lambda p}{}^{j}(k)\right]$$

(4.46)

$$\frac{\partial L}{\partial p(k)} = e_p{}^{j,J}(k) = \left[1 + \varphi_p(k) - \Delta t \frac{F(k)}{V(k)}\right]\left[\lambda_p{}^{j}(k) - \mu_p(k) e_{\lambda p}{}^{j}(k)\right]$$

(4.47)

$$\frac{\partial L}{\partial V(k)} = e_V{}^{j,J}(k) = \varphi_V(k) x(k)\left[\lambda_x{}^{j}(k) - \mu_x(k) e_{\lambda x}{}^{j}(k)\right] -$$
$$- \varphi_V(k)\left[s_i - s(k)\right]\left[\lambda_s{}^{j}(k) - \mu_s(k) e_{\lambda s}{}^{j}(k)\right] +$$
$$+ \varphi_V(k) p(k)\left[\lambda_p{}^{j}(k) - \mu_p(k) e_{\lambda p}{}^{j}(k)\right] + \left[\lambda_V{}^{j}(k) - \mu_V(k) e_{\lambda V}{}^{j}(k)\right]$$

(4.48)

For the control variables

$$\frac{\partial L}{\partial F(k)} = e_F{}^{j,J}(k) =$$
$$= \frac{\Delta t x(k)}{V(k)}\left[\lambda_x{}^{j}(k) - \mu_x(k) e_{\lambda x}{}^{j}(k)\right] + \frac{\Delta t[s_i - s(k)]}{V(k)}\left[\lambda_s{}^{j}(k) - \mu_s(k) e_{\lambda s}{}^{j}(k)\right] -$$
$$- \frac{\Delta t p(k)}{V(k)}\left[\lambda_p{}^{j}(k) - \mu_p(k) e_{\lambda p}{}^{j}(k)\right] + \Delta t\left[\lambda_V{}^{j}(k) - \mu_V(k) e_{\lambda V}{}^{j}(k)\right]$$

(4.49)

$$\frac{\partial L}{\partial T(k)} = e_T{}^{j,J}(k) =$$
$$= \left[\varphi_1(k)\frac{\partial \mu_{\max}}{\partial T} - \varphi_2(k)\frac{\partial K_s}{\partial T}\right]\left[\lambda_x{}^{j}(k) - \mu_x(k) e_{\lambda x}{}^{j}(k)\right] +$$

$$+\left[\left[-\varphi_1(k)\right]\left[\frac{\frac{\partial\mu_{\max}}{\partial T}Y_{x/s}-\mu_{\max}\frac{\partial Y_{x/s}}{\partial T}}{Y^2{}_{x/s}}\right]+\right.$$

$$\left.+\frac{\partial_2(k)}{Y_{x/s}}\frac{\partial K_s}{\partial T}\right]\left[\lambda_s{}^j(k)-\mu_s(k)e_{\lambda s}{}^j(k)\right]+\left[\varphi_1(k)\left[\frac{\partial Y_{p/x}}{\partial T}\mu_{\max}+Y_{p/x}\frac{\partial\mu_{\max}}{\partial T}\right]-\right.$$

$$\left.-Y_{p/x}\varphi_2(k)\frac{\partial K_s}{\partial T}\right]\left[\lambda_p{}^j(k)-\mu_p(k)e_{\lambda p}{}^j(k)\right] \qquad (4.50)$$

$$\frac{\partial L}{\partial pH}=e_{pH}{}^{jJ}(k)=$$

$$=\left[\varphi_1(k)\frac{\partial\mu_{\max}}{\partial pH}-\varphi_2(k)\frac{\partial K_s}{\partial pH}\right]\left[\lambda_x{}^j(k)-\mu_x(k)e_{\lambda x}{}^j(k)\right]+$$

$$+\left[\left[-\varphi_1(k)\right]\left[\frac{\frac{\partial\mu_{\max}}{\partial pH}Y_{x/s}-\mu_{\max}\frac{\partial Y_{x/s}}{\partial pH}}{Y^2{}_{x/s}}\right]+\right.$$

$$\left.+\frac{\partial_2(k)}{Y_{x/s}}\frac{\partial K_s}{\partial pH}\right]\left[\lambda_s{}^j(k)-\mu_s(k)e_{\lambda s}{}^j(k)\right]+\left[\varphi_1(k)\left[\frac{\partial Y_{p/x}}{\partial pH}\mu_{\max}+Y_{p/x}\frac{\partial\mu_{\max}}{\partial pH}\right]-\right.$$

$$\left.-Y_{p/x}\varphi_2(k)\frac{\partial K_s}{\partial pH}\right]\left[\lambda_p{}^j(k)-\mu_p(k)e_{\lambda p}{}^j(k)\right] \qquad (4.51)$$

$$\frac{\partial L}{\partial DO_2}=e_{DO_2}{}^{jJ}(k)=$$

$$=\left[\varphi_1(k)\frac{\partial\mu_{\max}}{\partial DO_2}-\varphi_2(k)\frac{\partial K_s}{\partial DO_2}\right]\left[\lambda_x{}^j(k)-\mu_x(k)e_{\lambda x}{}^j(k)\right]$$

$$+\left[\left[-\varphi_1(k)\right]\left[\frac{\frac{\partial\mu_{\max}}{\partial DO_2}Y_{x/s}-\mu_{\max}\frac{\partial Y_{x/s}}{\partial DO_2}}{Y^2{}_{x/s}}\right]+\right.$$

$$\left.+\frac{\partial_2(k)}{Y_{x/s}}\frac{\partial K_s}{\partial T}\right]\left[\lambda_s{}^j(k)-\mu_s(k)e_{\lambda s}{}^j(k)\right]+\left[\varphi_1(k)\left[\frac{\partial Y_{p/x}}{\partial DO_2}\mu_{\max}+Y_{p/x}\frac{\partial\mu_{\max}}{\partial DO_2}\right]-\right.$$

$$-Y_{p/x}\varphi_2(k)\frac{\partial K_s}{\partial DO_2}\Big]\Big[\lambda_p{}^j(k)-\mu_p(k)e_{\lambda_p}{}^j(k)\Big] \tag{4.52}$$

The calculation of the gradients is done with the old values of the state and control variables $v^{j,i}$, $v=x,s,p,V,F,T,pH,DO_2$. The calculation with the gradient procedure will continue until the necessary conditions for optimality towards state and control variables are fulfilled. This means that the values of the gradients are approximately equal to zero:

$$\|e_v(k)\| \le \varepsilon_v, \varepsilon_v > 0, v=x,s,p,V,F,T,pH,DO_2, k=\overline{0,K-1} \tag{4.53}$$

The new calculated values have to be checked for belonging to the area of constraints (4.54)-(4.60).

$$T_{\min} \le T(k) \le T_{\max} \tag{4.54}$$

$$pH_{\min} \le pH(k) \le pH_{\max} \tag{4.55}$$

$$DO_{2\min} \le DO_2(k) \le DO_{2\max} \tag{4.56}$$

$$V_{\min} \le V(k) \le V_{\max} \tag{4.57}$$

$$F_{\min} \le F(k) \le F_{\max} \tag{4.58}$$

$$p_{\min} \le p(k) \le p_{\max} \tag{4.59}$$

$$s_{\min} \le s(k) \le s_{\max}, \quad k=\overline{0,K-1} \tag{4.60}$$

Simple way to do and correct the obtained new values is to project them over the constraints domain according to the relation:

$$v^{j,i}(k)=\begin{Bmatrix} v_{\min}, \text{ if } v^{j,i}(k) < v_{\min} \\ v^{j,i}(k), \text{ if } v_{\min} \le v^{j,i}(k) \le v_{\max}, \ k=\overline{0,K-1} \\ v_{\max} \text{ if } v^{j,i}(k) > v_{\max} \end{Bmatrix}$$

$$\text{where } v=x,s,p,V,F,T,pH,DO_2 \tag{4.61}$$

The obtained values of the trajectories of the state and control variables are function of the coordinating variables. They are optimal for the given values of the coordinating variables. If the coordinating trajectories are optimal ones, then the obtained state and control

variables also will be the optimal according to the theory of the duality in optimal control and Lagrange methods.

4.1.2.4 The coordinating sub-problem

The coordinating variables are obtained from the necessary conditions for optimality of the augmented Lagrange functional according to them. For the conjugate variables these are equations are (4.19)-(4.22). The necessary conditions for optimality according to the new introduced interconnections in time domain are:

$$\frac{\partial L}{\partial \delta_x(k)} = \mu_x\left[\delta_x(k) - f_x(k)\right] = e_{\delta x}(k) = 0 = \mu_x e_{\lambda_x}(k), \tag{4.62}$$

$$\frac{\partial L}{\partial \delta_s(k)} = \mu_s\left[\delta_s(k) - f_s(k)\right] = e_{\delta s}(k) = 0 = \mu_s e_{\lambda_s}(k), \tag{4.63}$$

$$\frac{\partial L}{\partial \delta_p(k)} = \mu_p\left[\delta_p(k) - f_p(k)\right] = e_{\delta p}(k) = 0 = \mu_p e_{\lambda p}(k), \tag{4.64}$$

$$\frac{\partial L}{\partial \delta_V(k)} = \mu_V\left[\delta_V(k) - f_V(k)\right] = e_{\delta V}(k) = 0 = \mu_p e_{\lambda V}(k), \tag{4.65}$$

$$where \delta_v(k) - f_v(k) = e_{\lambda_v}(k), v = x, s, p, V, \tag{4.66}$$

according to equations (4.19)-(4.22).

The necessary conditions for optimality (4.19)-(4.22) and (4.62)-(4.66) can be combined in a gradient procedure. Equations (4.62)-(4.66) give the values of the gradients for the conjugate variables multiplied by the penalty co-efficient. μ_v, $v = x, s, p, V$.

It appears that:

$$\frac{\partial L}{\partial \delta_v(k)} = \mu_v \frac{\partial L}{\partial_{\lambda_v}(k)} = \mu_v e_{\lambda_v}(k), \quad k = \overline{0, K-1} \tag{4.67}$$

Gradient procedure can be built for both type of coordinating variables.

For conjugate variables:

$$\lambda_v^{j+1}(k) = \lambda_v^{j}(k) - \alpha_{\lambda_v} e^{j}{}_{\lambda v}(k), \tag{4.68}$$

$$e^{j}{}_{\lambda v}(k) = \delta_v^{j}(k) - f^{j}{}_{v}(k), v = x, s, p, V, k = \overline{0, K-1}, \tag{4.69}$$

$$\delta_v^{j+1}(k) = \delta_v^{j}(k) + \alpha_{\delta v} e_{\delta v}(k) = \delta_v^{j}(k) + \alpha_{\delta v} \mu_v e_v(k),\ k = \overline{0, K-1} \tag{4.70}$$

$v = x, s, p, V$

where $\alpha_{\lambda v}$ and $\alpha_{\delta v}$ are the steps of the procedures for calculation of the conjugate variables and the interconnection in time, respectively.

As the calculations are based on gradient iterative procedure, they will stop when the gradient is going to zero, or practically achieving some conditions for optimality, given by

$$\left\| e_{\lambda v}(k) \right\| \le \varepsilon_{\lambda v}, \varepsilon_{\lambda v} > 0, k = \overline{0, K-1}, \tag{4.71}$$

where $\varepsilon_{\lambda v}$ is a very small positive number.

For every step of the gradient procedure for the coordinating process the full iteration cycle of the sub problems of the first level is done. The calculations of the coordinating variables are done with the values of state and control variables of the first level sub problems and the calculations of the first level sub problems are done with the values of the calculated on the previous iteration coordinating variables. The iterative process of coordination and the optimal solution of the whole problem for optimal control is obtained when the optimal solution of the coordinating sub problem is reached, this means conditions (4.71) are fulfilled.

4.2 Program for solution of the optimal control problem in Matlab

4.2.1 Algorithm of the method

The calculations in the two-level structure can be grouped and represented in the following algorithm:

On the coordinating level the initial values of the coordinating and gradient procedures, initial trajectories of the state, control and coordinating variables are set as follows:

a) Maximal number of step of the coordinating process – M2.
   Maximal number of steps for state and control variables gradient procedure – M1.

b) Steps of the gradient procedures

i. For coordinating conjugate variables - $\alpha_{\lambda v}, v = x, s, p, V$.

ii. For coordinating interconnection in time - $\alpha_{\delta v}, v = x, s, p, V$.

iii. For state variables and control variables

$\alpha_v, v = x, s, p, V, F, T, pH, DO_2$

c) Values for error and termination of the calculations for coordinating procedures. $\varepsilon_{\lambda v}, \varepsilon_{\delta v}, v = x, s, p, V$ and for gradient procedures on first level. $\varepsilon_v, v = x, s, p, V$

d) Initial values trajectories of the conjugate variables.

$\lambda^1{}_v(k), v = x, s, p, V, k = \overline{0, K-1}$

e) Initial values of the state variables, x(0), s(0), p(0), V(0).

f) Number of steps in the optimization horizon – K.

g) Initial trajectories of the control variables

$F^{1,1}(k), T^{1,1}(k), pH^{1,1}(k), DO_2{}^{1,1}(k),\ k = \overline{0, K-1}$,

h) The initial trajectories of the control variables are substituted in the model equations (4.2)-(4.9) and the state trajectories are calculated.

i) The initial trajectories of the interconnections in time domain are calculated from

$\delta_v(k) = v(k+1), k = \overline{0, K-1}$

j) Initial trajectories of the coordinating variables and initial state and control trajectories are sent to the first level.

2. On the first level the following calculations are done:

a. Derivatives from equation (4.27)-(4.29) are calculated.

b. The kinetic coefficients $\mu_{\max}, K_s, Y_{x/s}, Y_{p/x}$ from equations (4.6)-(4.9) are calculated.

c. The expressions $\varphi_x(k), \varphi_s(k), \varphi_p(k), \varphi_V(k), \varphi_1(k), \varphi_2(k), k = \overline{0, K-1}$ are calculated from equations (4.39)-(4.44).

d. The gradients $e_v{}^{j,i}(k), k = \overline{0, K-1}, v = x, s, p, V, F, T, pH, DO_2$ are calculated from equations (4.45)-(4.52).

e. The errors for the end of the gradient procedures are calculated according to equation (4.53). If the error is less than ε_v, the calculations of the first level stop and the current values of the state and control variables are the optimal ones. If the error conditions are not satisfied the new values of the state and control values are calculated.

f. The new values of state and control variables are calculated from equation (4.30).

g. The obtained state and control variables are projected over the constraint domain according to equation (4.61). The obtained trajectories are sent to the second level.

3. On the second level the following calculations are done:

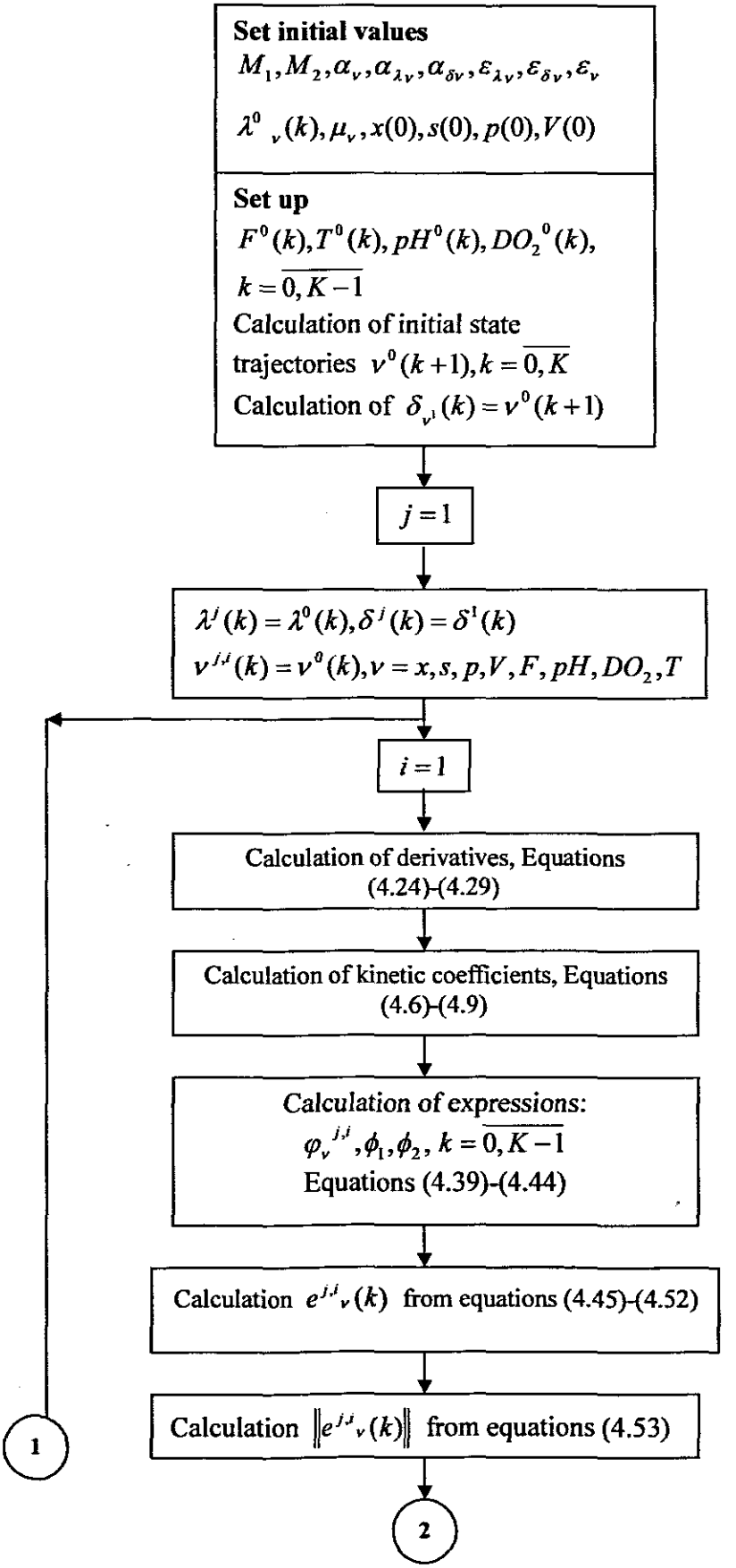
h. The gradients $e_{\lambda_\nu}(k), k=\overline{0,K-1}, \nu = x,s,p,V$ are calculated from equations (4.19)-(4.22).

i. The conditions for the end of the calculations are checked according to Equation (4.71). If the conditions are fulfilled the optimal solution of the coordinating and of the whole problem are obtained. If the conditions are not satisfied the new values of the coordinating variables are calculated.

j. New values of the conjugate variables are calculated according to equation (4.68)-(4.69).

k. The new values of the interconnections are calculated from equation (4.70). These trajectories are sent to the first level sub problems where the calculations from p.2a till 3b are repeated and so on.

The schematic diagram of the algorithm is given on Fig 4.2.


Set initial values
$M_1, M_2, \alpha_\nu, \alpha_{\lambda\nu}, \alpha_{\delta\nu}, \varepsilon_{\lambda\nu}, \varepsilon_{\delta\nu}, \varepsilon_\nu$
$\lambda^0{}_\nu(k), \mu_\nu, x(0), s(0), p(0), V(0)$
Set up
$F^0(k), T^0(k), pH^0(k), DO_2{}^0(k)$,
$k = \overline{0, K-1}$
Calculation of initial state
trajectories $\nu^0(k+1), k = \overline{0, K}$
Calculation of $\delta_{\nu^1}(k) = \nu^0(k+1)$
$j = 1$
$\lambda^j(k) = \lambda^0(k), \delta^j(k) = \delta^1(k)$
$\nu^{j,i}(k) = \nu^0(k), \nu = x, s, p, V, F, pH, DO_2, T$
$i = 1$
Calculation of derivatives, Equations (4.24)-(4.29)
Calculation of kinetic coefficients, Equations (4.6)-(4.9)
Calculation of expressions:
$\varphi_\nu{}^{j,i}, \phi_1, \phi_2$, $k = \overline{0, K-1}$
Equations (4.39)-(4.44)
Calculation $e^{j,i}{}_\nu(k)$ from equations (4.45)-(4.52)
Calculation $\|e^{j,i}{}_\nu(k)\|$ from equations (4.53)
1
2

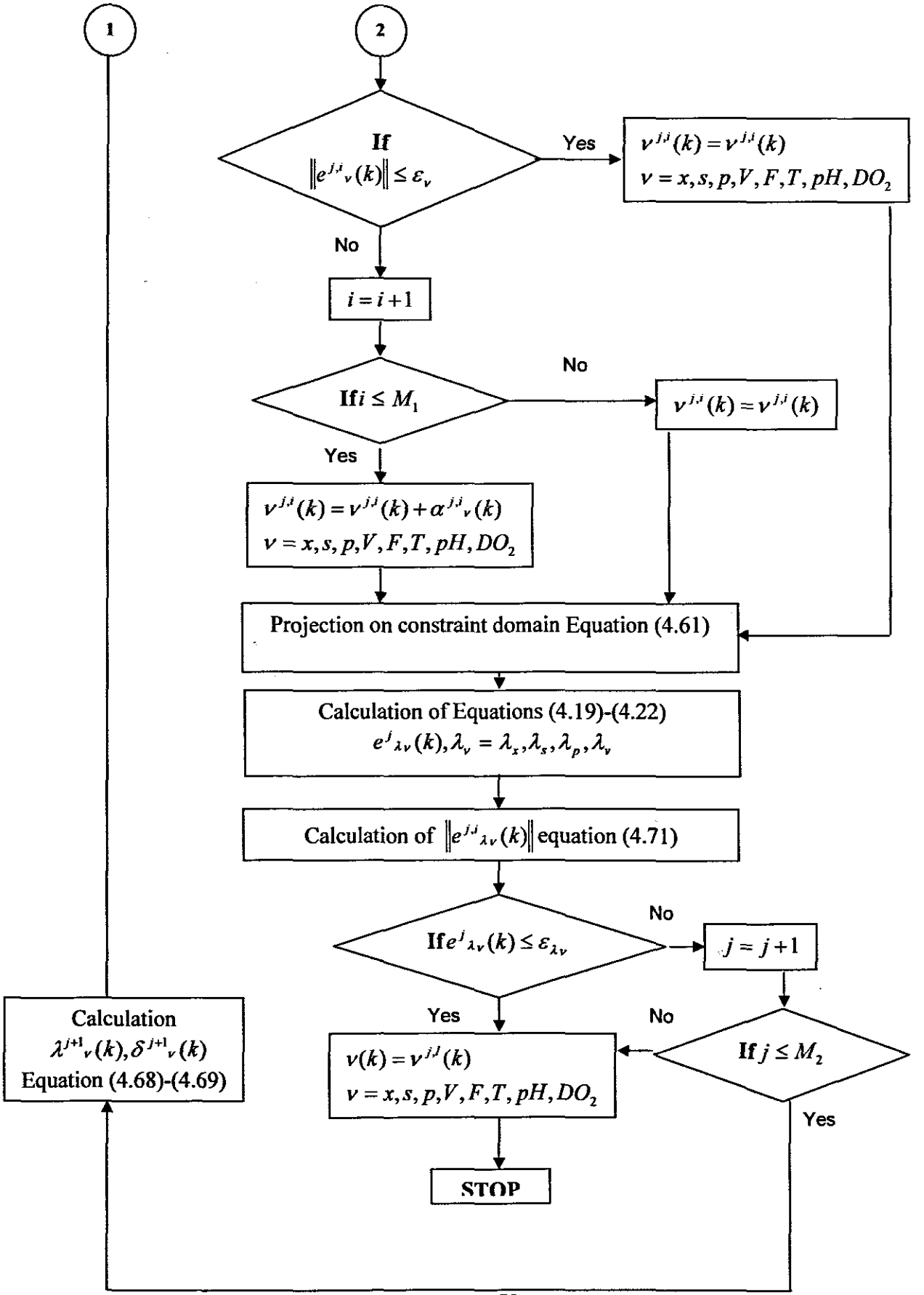

1
2
If $\|e^{j,i}{}_{v}(k)\| \le \varepsilon_v$
Yes
$v^{j,i}(k) = v^{j,i}(k)$
$v = x, s, p, V, F, T, pH, DO_2$
No
$i = i + 1$
If $i \le M_1$
No
$v^{j,i}(k) = v^{j,i}(k)$
Yes
$v^{j,i}(k) = v^{j,i}(k) + \alpha^{j,i}{}_{v}(k)$
$v = x, s, p, V, F, T, pH, DO_2$
Projection on constraint domain Equation (4.61)
Calculation of Equations (4.19)-(4.22)
$e^{j}{}_{\lambda v}(k), \lambda_v = \lambda_x, \lambda_s, \lambda_p, \lambda_v$
Calculation of $\|e^{j,i}{}_{\lambda v}(k)\|$ equation (4.71)
If $e^{j}{}_{\lambda v}(k) \le \varepsilon_{\lambda v}$
No
$j = j + 1$
Yes
No
If $j \le M_2$
Yes
Calculation
$\lambda^{j+1}{}_{v}(k), \delta^{j+1}{}_{v}(k)$
Equation (4.68)-(4.69)
$v(k) = v^{j,i}(k)$
$v = x, s, p, V, F, T, pH, DO_2$
STOP

Fig 4.2 Algorithm for calculation of the optimal state and control trajectories in a sequential way

4.2.2. Program description

The program will be described in chapter 6.

4.3 Results from the calculations

The results from the calculations also will be described in chapter 6.

4.4 Conclusion

This chapter describes the optimal control problem is solved using optimal control theory, the approach of the functional of Lagrange. The inputs, outputs, functions and results of the programmes for sequential optimal control calculation in Matlab/Simulink will be given in the Chapter 6. Next chapter will describe the concepts and evolution of parallel and distributed computing, the distributed computing toolbox and MATLAB distributed computing Engine, the building of the distributed computing programmes.

Chapter 5

Parallel and distributed computing in a cluster computers

5. Introduction

This chapter describes the concepts and evolution of parallel and distributed computing. The distributed computing toolbox and MATLAB distributed computing Engine are introduced. The building of the distributed computing programmes is described.

5.1. Parallel Processing

5.1.1 Parallel Processing: Concepts and Evolution

In day-to-day life, people are required to complete many tasks within a limited timespan. Speed is the main issue of concern with such time-bound problems. Moreover, performance, efficiency and the smooth running of any system are highly dependent on speed. Of the mechanisms that are generally adopted to increase performance speed, parallel processing is most worthy of mention. It is utilized in many real-life systems. The parallel processing can be simply defined as completing a large task quicker and more efficiently by dividing it into several smaller tasks and executing them simultaneously using more resources. However, while applying parallel processing to a process, a number of factors are required to be considered, for example, whether the task in question can be performed in parallel, its cost effectiveness, synchronization of the subtasks, and communication among the resources. Practically, tasks that can effectively be divided are good candidates for parallel processing. Speed is probably the most frequently addressed term in computing. The very purpose of the invention and evolution of the microprocessor is to add speed to the normal flow of human civilization. Like all other cases, speed is the main issue of concern in the computational world, mainly due to real-time requirements. Many scientific and engineering problems are required to be completed within specific time periods to produce accurate and valid results. They often involve huge repetitive calculations on large amounts of data (Tokhi et al., 2002).

5.1.2 Basic Components of Parallel Processing

Parallelism enables multiple processors to work simultaneously on several parts of a program in order to execute it faster than could be done otherwise. As such, parallel processing entails some basic components for implementation.

1 Computer hardware that supports parallelism. This hardware is designed to work with multiple processors and may all reside either on the same computer or may be spread across several computers. When spread across several computers, each computer is referred to as a node. A means of communication among the processors or nodes is required for message transmission for synchronization of various parts of the program to ensure correctness of the final result.

2 Operating system and application software that are capable of managing multiple processors.

Parallel Processing: Tasks, Processes, Processors and Basic Characteristics

To understand parallel processing at the implementation level, one must know some basic terms such as task, process, processors and the fundamental characteristics of parallel processing. These are described below (Tokhi et al., 2002).

Task

A task is the smallest unit of the work that needs to be done. In other words, it is the independently executable smallest unit of a program that is executed by one processor and concurrency among processors is exploited only across tasks. Tasks can be of two types, namely fine-grained and coarse-grained. If the amount of work a task performs is small it is called a fine-grained task and otherwise it is called a coarse-grained task.

Process

A process performs the tasks assigned to it. Usually more than one task is assigned to a process. However, a process can consist of only one task. A program in the parallel executable format is composed of a number of processes, each of which performs a subset of tasks in the program. Processes usually need to communicate and synchronize with one another to perform tasks assigned to them by executing them on the physical processors in

the machine.

Processor

While the task and process are software-based entities, the processor is a hardware-based physical resource. A processor executes the process assigned to it. Usually a parallel program is written in terms of processors, and the number of processes in the program could be the same as the number of processors, or higher or less than the number of processors in the parallel architecture. If the number of processes is more than the number of processors, more than one process is assigned to some or all of the processors, depending on the number of processes and nature of the algorithm. If the number of processes is less than the number of processors, some processors are not assigned processes and they remain idle during execution of the program.

Basic Characteristics

A parallel processing system has the following characteristics:

1 Each processor in a system can perform tasks concurrently. They might have their own memory units for executing and managing the tasks and/or can share a common memory unit.

2 Tasks may need to be synchronized. Sometimes synchronization is vital to ensure correctness of the results.

3 Processors or nodes usually share resources, such as data, disks, and other devices.

4 A parallel system executes tasks or programs faster than sequential processing of the same tasks or programs.

5.1.3 Parallel Processing: Advantages

Parallel processing offers several advantages over sequential processing. These are described below (Tokhi et al., 2002).

Speedup

Parallel processing reduces execution time of the program considered and hence increases speedup or improves response time. Speedup is defined as the ratio of the execution time

with one processor to the execution time using a number of processors, and is calculated as:

$$Speedup = \frac{T_1}{T_n} \qquad (5.1)$$

where, T_1 is the time taken by one processor to execute a task and T_n is the time taken by n processors to execute the same task.

The ideal speedup curve is seldom achieved due to such factors as communication overhead and memory access. It is rarely possible to parallelize an algorithm without communication among the processors involved. On the other hand, some programs can easily be divided into parts and allocated to processors whereas others do not lend themselves to this mechanism.

Scaleup

Scaleup or high throughput of a system can be defined as the ability to maintain a constant execution time period as the workload or job size increases by adding additional hardware such as processors, memory and disks. While calculating the speedup, the problem size is kept fixed whereas scaleup is calculated by increasing the workload size or transaction volume. Scaleup is measured in terms of how much transaction volume can be increased by adding more hardware, specially processors, while still maintaining constant execution time. Scaleup is calculated by:

$$Scaleup = \frac{W_N}{W_1} \qquad (5.2)$$

where W_1 is workload or transaction volume executed in a certain amount of time using one processor and W_N is the workload executed in the same time using n processors.

Parallel systems provide much better scalability in comparison to single-processor systems as added workload or transaction volume over time can easily be handled by adding extra processing power (especially processors) without loss of response time.

Fault Tolerance

A parallel processing system is more likely to be fault tolerant than a single-processor system, as parallel systems offer the flexibility of adding and removing processors or other

additional hardware in the system. For example, if a processor in a parallel system does not work for some reason it could easily be replaced with another one without affecting the ongoing work. This is not possible in the case of a single-processor system.

Cost-to-Performance Ratio

The design and development of a faster computer increases costs significantly. Accordingly, increasing the processing power on a single processor becomes technically difficult and very expensive beyond a certain limit. In such cases parallel systems provide better performance for the price.

Handling Larger Tasks

It is easily understood that a parallel system can handle bigger problems than a single-processor system. If the problem tends to grow over time, parallel processing can cope with it by adding additional hardware.

5.1.4 Parallel Processing: Some Applications

Parallel processing has a wide range of applications in various fields that range from weather forecasting to robotics. Some of the major applications of parallel processing are: Weather Forecasting; Motion of Astronomical Bodies; Database Management, Satellite, Radar and Sonar Applications; Aerospace Applications; Robotics Applications; Signal Processing and Control.

Parallel processing is becoming increasingly important in various applications. It is utilized in many fields for better speedup and scaleup. To gain maximum benefit from a parallel solution requires various issues to be considered. It is always necessary to do some kind of background study and investigation on the basis of the factors involved to find whether parallel processing is worthwhile for a particular case. Currently, parallel processing is used extensively in various applications. Among all such applications signal processing and control demand special attention due to their general applicability in various fields and the current demand for automation (Tokhi et al., 2002).

5.1.5 PDS taxonomy

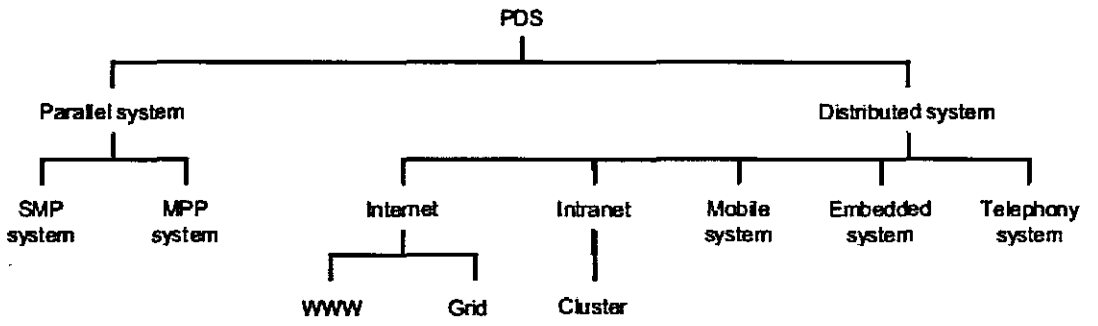


Fig 5.1 PDS taxonomy

The categorization of *PDS taxonomy* is shown in Figure 5.1. A *parallel system* consists of multiple processors in close communication, normally located within the same machine. Multiple processors can share the same memory or access their own local memory. A parallel system provides increased throughput and reliability (Sulistio *et al.*, 2004).

A parallel system can be classified into symmetric multiprocessing (SMP) systems and massively parallel processing (MPP) systems. A SMP system has multiple processors sharing a common memory and operating system. A MPP system has multiple processors accessing their own local memory and operating system. A MPP system is more difficult to program for parallelism compared with a SMP system since each processor accesses each local memory separate from other processors. However, a MPP system is more scalable in terms of the number of processors, since a SMP system is limited to a maximum number of processors.

A *distributed system* spreads out the computation among autonomous computers, which are physically distributed in general. Each distributed computer accesses its own local memory and communicates with other computers through networks such as the Internet. A distributed system supports both resource sharing and load sharing, and is fault-tolerant through the replication of devices and data in separate physical locations. It also provides a good price/performance ratio.

A distributed system can be further sub-divided into five types: Internet, intranet, mobile systems, embedded systems or telephony systems. The Internet is a heterogeneous network of machines and applications that is implemented using the Internet protocol (IP). The

World Wide Web (WWW) is built on top of the Internet architecture to share information, identified through a uniform resource locator (URL). A Grid (Foster and Kesselman, 1999) is a very large-scale distributed system that can scale to Internet-size environments with machines distributed across multiple organizations and administrative domains.

An Intranet is a local area network (LAN) that is usually privately owned by an organization. Access of the intranet is internal within the organization and firewalls guard the interface for external access to the Internet. A cluster (Barker and Buyya, 1999) is a collection of distributed machines that collaborate to present a single integrated computing resource to the user.

A mobile system is implemented using wireless network protocol so that it can function while it is on the move at different physical locations. Some examples of mobile systems are cellular phone systems, handheld devices and laptop computers with a wireless LAN.

An embedded system is a combination of hardware and software designed to function as a type of application device. Some examples of embedded systems are industrial machines, medical equipments, automobiles and aircrafts. A telephony system transmits voice and data using digital transmission through telephone networks. Some examples of telephony systems are asymmetric digital subscriber line (ADSL), integrated services digital network (ISDN), intelligent networks and advanced intelligent networks.

5.2 Designing and Building Parallel Programs

Designing and Building Parallel Programs promotes a view of parallel programming as an engineering discipline, in which programs are developed in a methodical fashion and both cost and performance are considered in a design (Foster, 1995).

A parallel machine model called the *multicomputer* fits these requirements. As illustrated in Figure 5.2, a multicomputer comprises a number of von Neumann computers, or *nodes*, linked by an *interconnection network*. Each computer executes its own program. This program may access local memory and may send and receive messages over the network. Messages are used to communicate with other computers or, equivalently, to read and write remote memories. In the idealized network, the cost of sending a message between two nodes is independent of both node location and other network traffic, but does depend on

message length.

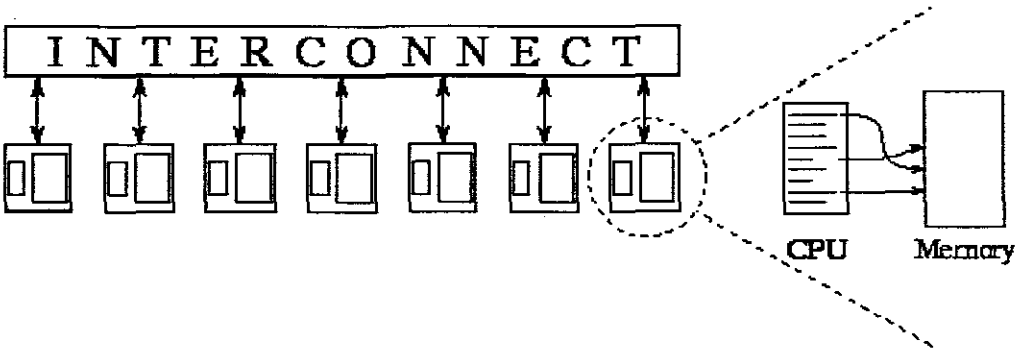


Figure 5.2: *The multicomputer, an idealized parallel computer model. Each node consists of a von Neumann machine: a CPU and memory. A node can communicate with other nodes by sending and receiving messages over an interconnection network.*

A defining attribute of the multicomputer model is that accesses to local (same-node) memory are less expensive than accesses to remote (different-node) memory. That is, read and write are less costly than send and receive. Hence, it is desirable that accesses to local data be more frequent than accesses to remote data. This property, called *locality*, is a third fundamental requirement for parallel software, in addition to concurrency and scalability. The importance of locality depends on the ratio of remote to local access costs. This ratio can vary from 10:1 to 1000:1 or greater, depending on the relative performance of the local computer, the network, and the mechanisms used to move data to and from the network.

Tasks and Channels

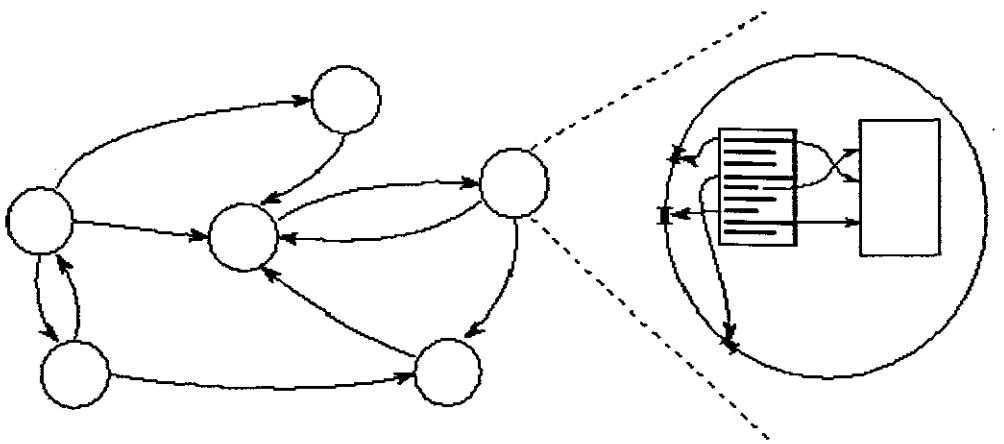


Figure 5.3: *A simple parallel programming model. The figure shows both the instantaneous state of a computation and a detailed picture of a single task. A computation consists of a set of tasks (represented by circles) connected by channels (arrows). A task encapsulates a program and local memory and defines a set of ports that define its interface to its environment. A channel is a message queue into which a sender can place messages and from which a receiver can remove messages, ``blocking'' if messages are not available.*

We consider next the question of which abstractions are appropriate and useful in a parallel programming model. Clearly, mechanisms are needed that allow explicit discussion about concurrency and locality and that facilitate development of scalable and modular programs. Also needed are abstractions that are simple to work with and that match the architectural model, the multicomputer. While numerous possible abstractions could be considered for this purpose, two fit these requirements particularly well: the *task* and *channel*. These are illustrated in Figure 5.3 and can be summarized as follows (Foster, 1995):

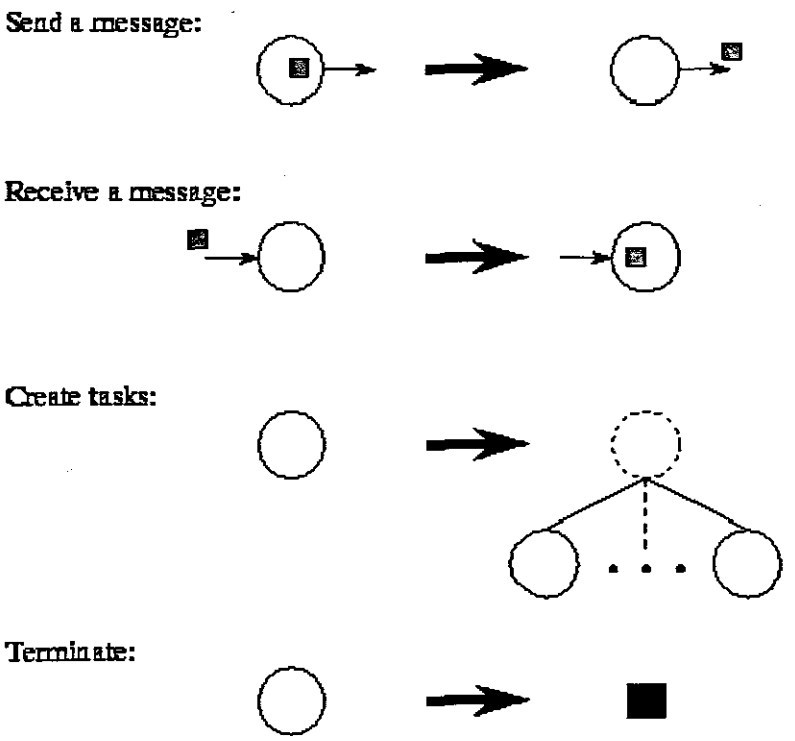


Figure 5.4: *The four basic task actions. In addition to reading and writing local memory, a task can send a message, receive a message, create new tasks (suspending until they terminate), and terminate.*

1. A parallel computation consists of one or more tasks. Tasks execute concurrently. The number of tasks can vary during program execution.
2. A task encapsulates a sequential program and local memory. (In effect, it is a virtual von Neumann machine.) In addition, a set of *inports* and *outports* define its interface to its environment.
3. A task can perform four basic actions in addition to reading and writing its local memory (Figure 5.4): send messages on its outports, receive messages on its inports, create new tasks, and terminate.
4. A send operation is asynchronous: it completes immediately. A receive operation is synchronous: it causes execution of the task to block until a message is available.
5. Outport/inport pairs can be connected by message queues called *channels*. Channels can be created and deleted, and references to channels (ports) can be included in messages, so connectivity can vary dynamically.
6. Tasks can be mapped to physical processors in various ways; the mapping employed does not affect the semantics of a program. In particular, multiple tasks can be mapped to a single processor. (We can also imagine a single task being mapped to multiple processors, but that possibility is not considered here.)

The task abstraction provides a mechanism for talking about locality: data contained in a task's local memory are ``close"; other data are ``remote." The channel abstraction provides a mechanism for indicating that computation in one task requires data in another task in order to proceed. (This is termed a *data dependency*).

This task/channel programming model has some other properties: performance, mapping independence, modularity, and determinism.

Performance. Sequential programming abstractions such as procedures and data structures are effective because they can be mapped simply and efficiently to the von Neumann computer. The task and channel have a similarly direct mapping to the multicomputer. A task represents a piece of code that can be executed sequentially, on a single processor. If

two tasks that share a channel are mapped to different processors, the channel connection is implemented as interprocessor communication; if they are mapped to the same processor, some more efficient mechanism can be used.

Mapping Independence. Because tasks interact using the same mechanism (channels) regardless of task location, the result computed by a program does not depend on where tasks execute. Hence, algorithms can be designed and implemented without concern for the number of processors on which they will execute; in fact, algorithms are frequently designed that create many more tasks than processors. This is a straightforward way of achieving *scalability*: as the number of processors increases, the number of tasks per processor is reduced but the algorithm itself need not be modified. The creation of more tasks than processors can also serve to mask communication delays, by providing other computation that can be performed while communication is performed to access remote data.

Modularity. In modular program design, various components of a program are developed separately, as independent modules, and then combined to obtain a complete program. Interactions between modules are restricted to well-defined interfaces. Hence, module implementations can be changed without modifying other components, and the properties of a program can be determined from the specifications for its modules and the code that plugs these modules together. When successfully applied, modular design reduces program complexity and facilitates code reuse.

Strong similarities exist between the task/channel model and the popular object-oriented programming paradigm. Tasks, like objects, encapsulate data and the code that operates on those data. Distinguishing features of the task/channel model are its concurrency, its use of channels rather than method calls to specify interactions, and its lack of support for inheritance.

Determinism. An algorithm or program is deterministic if execution with a particular input always yields the same output. It is nondeterministic if multiple executions with the same input can give different outputs. Although nondeterminism is sometimes useful and must be supported, a parallel programming model that makes it easy to write deterministic programs is highly desirable. Deterministic programs tend to be easier to understand. Also, when checking for correctness, only one execution sequence of a parallel program

needs to be considered, rather than all possible executions (Foster, 1995).

Task-scheduling algorithms

Algorithms based on functional decomposition often yield computations consisting of many short-lived tasks that coordinate with other tasks only at the start and end of execution. We can use *task-scheduling* algorithms, which allocate tasks to processors that are idle or that are likely to become idle.

Task-scheduling algorithms can be used when a functional decomposition yields many tasks, each with weak locality requirements. A centralized or distributed task pool is maintained, into which new tasks are placed and from which tasks are taken for allocation to processors. In effect, we reformulate the parallel algorithm so that what were originally conceived of as tasks become data structures representing ``problems," to be solved by a set of worker tasks, typically one per processor.

The most critical (and complicated) aspect of a task-scheduling algorithm is the strategy used to allocate problems to workers. Generally, the chosen strategy will represent a compromise between the conflicting requirements for independent operation (to reduce communication costs) and global knowledge of computation state (to improve load balance). We discuss manager/worker, hierarchical manager/worker, and decentralized approaches (Foster, 1995).

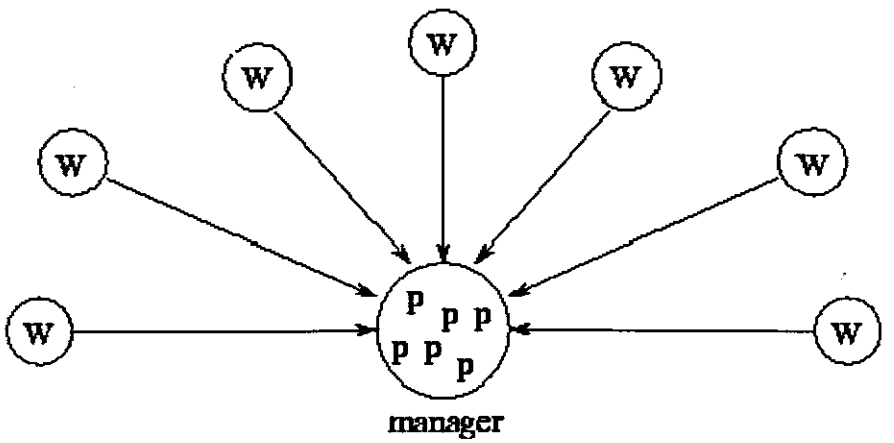


Figure 5.5: *Manager/worker load-balancing structure. Workers repeatedly request and process problem descriptions; the manager maintains a pool of problem descriptions (p) and responds to requests from workers.*

Manager/Worker

Figure 5.5 illustrates a particularly simple task scheduling scheme that is nevertheless effective for moderate numbers of processors. A central manager task is given responsibility for problem allocation. Each worker repeatedly requests and executes a problem from the manager. Workers can also send new tasks to the manager for allocation to other workers. The efficiency of this strategy depends on the number of workers and the relative costs of obtaining and executing problems. Efficiency can be improved by prefetching problems so as to overlap computation and communication, and by caching problems in workers, so that workers communicate with the manager only when no problems are available locally.

Hierarchical Manager/Worker

A variant of the manager/worker scheme divides workers into disjoint sets, each with a submanager. Workers request tasks from submanagers, which themselves communicate periodically with the manager and with other submanagers to balance load between the sets of processors for which they are responsible.

Decentralized Schemes

In completely decentralized schemes, there is no central manager. Instead, a separate task pool is maintained on each processor, and idle workers request problems from other processors. In effect, the task pool becomes a distributed data structure that is accessed by the different tasks in an asynchronous fashion. A variety of access policies can be defined. For example, a worker may request work from a small number of predefined ``neighbors" or may select other processors at random. In a hybrid centralized/distributed scheme, requests are sent to a central manager, which allocates them to workers in a round-robin fashion. Notice that while this manager will certainly be a bottleneck on large numbers of processors, it will typically be accessed less frequently than will the manager in a manager/worker scheduler and hence is a more scalable constructed.

Access to a distributed data structure, such as the task pool maintained by a decentralized load-balancing scheme, can be provided in several different ways. Workers can be made responsible for both computing and managing the queue of problems. In this case, each

worker must periodically poll to detect pending requests. Alternatively, computation and task pool management responsibilities can be encapsulated in separate tasks.

Termination Detection

Task-scheduling algorithms require a mechanism for determining when a search is complete; otherwise, idle workers will never stop requesting work from other workers. This *termination detection* operation is straightforward in centralized schemes, because the manager can easily determine when all workers are idle. It is more difficult in decentralized algorithms, because not only is there no central record of which workers are idle, but also messages in transit may be carrying tasks even when all workers appear to be idle. See the chapter notes for references to termination-detection algorithms.

Modularity and Parallel Computing

The design principles reviewed in the preceding section apply directly to parallel programming. However, parallelism also introduces additional concerns. A sequential module encapsulates the code that implements the functions provided by the module's interface and the data structures accessed by those functions. In parallel programming, we need to consider not only code and data but also the tasks created by a module, the way in which data structures are partitioned and mapped to processors, and internal communication structures. Probably the most fundamental issue is that of data distribution.

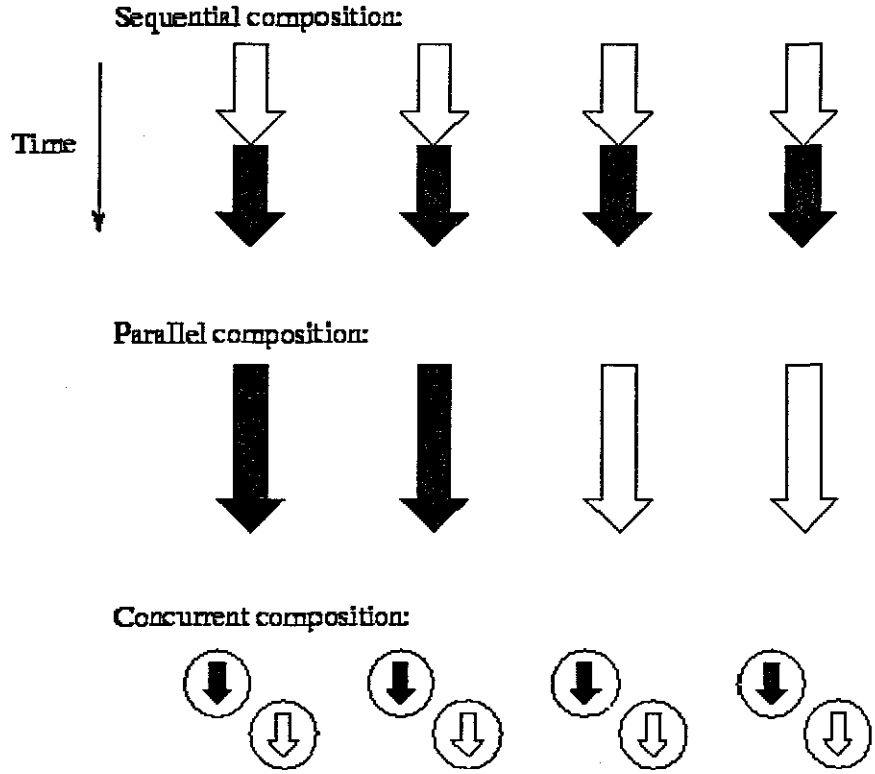


Figure 5.6: *Three forms of parallel program composition. In each case, the program is shown executing on four processors, with each arrow representing a separate thread of control and shading denoting two different program components. In sequential composition, different program components execute in sequence on all processors. In parallel composition, different program components execute concurrently on different processors. In concurrent composition, different program components execute concurrently on the same processors.*

Another difference between sequential and parallel programming is that in the former, modules can be put together (composed) in just one way: sequentially. Execution of a program leads to a sequence of calls to functions defined in different modules. This is called *sequential composition* and can also be used in parallel programming, and indeed is fundamental to the Sport Psychology Movie Database programming model used in many parallel programs. However, we often need to compose program components in other ways (Figure 5.6). In *parallel composition*, different modules execute concurrently on disjoint sets of processors. This strategy can enhance modularity and improve scalability and locality. In *concurrent composition*, different modules execute concurrently on the same

processors, with execution of a particular module enabled by the availability of data. Concurrent composition can both reduce design complexity and allow overlapping of computation and communication.

We distinguish between sequential, parallel, and concurrent composition both because they are different ways of thinking about programs and because not all parallel programming tools support all three compositional forms. Data-parallel languages (such as High Performance Fortran) tend to support only sequential composition. Message-passing libraries (such as Message Passing Interface) typically support both sequential and parallel composition but not concurrent composition. Other languages and libraries (such as CC++ and Fortran M) support all three forms of composition (Foster, 1995).

Data Distribution

The distribution of a program's data structures among tasks and processors (that is, the way in which data structures are partitioned and mapped) is an important aspect of parallel algorithm design. We also knew how to design data distributions that maximize performance and/or minimize software engineering costs.

Data distribution can become a more complex issue in programs constructed from several components. Simply choosing the optimal distribution for each component may result in different modules using different data distributions. For example, one module may output an array data structure distributed by columns, while another expects its input to be distributed by rows. If these two modules are to be composed, then either the modules themselves must be modified to use different distributions, or data must be explicitly redistributed as they are passed from one component to the other. These different solutions can have different performance characteristics and development costs.

Both performance tuning and program reuse are made easier if modules are designed to be *data distribution neutral* , that is, if they can deal with a variety of different data distributions. This neutrality can be achieved by specifying the distribution of a particular data structure as a runtime parameter or in the data structure itself. For example, the two modules referred to in the preceding paragraph could be defined to deal with arbitrary two-dimensional decompositions. The combined program could then utilize a decomposition by rows, a decomposition by columns, or (as a compromise) a

two-dimensional decomposition.

Designing a module to be data distribution neutral is not necessarily easy. In some cases, different data distributions may call for quite different algorithms.

Sequential Composition

In a parallel program constructed using only sequential composition, each processor inevitably executes the same program, which in turn performs a series of calls to different program components. These program components may themselves communicate and synchronize, but they cannot create new tasks. Hence, the entire computation moves sequentially from one parallel operation to the next.

As an example, consider the following program, which could be executed by each task in an SPMD finite difference program.

```
while (not done) do
      finite_difference(localgrid, localmax)
      global_maximum(localmax, globmax)
      if(globmax < threshold) done = true
enddo
```

This program is structured as a sequential composition of two procedure calls and a conditional statement. At each step, each task first calls the procedure `finite_difference` to advance the simulation on its part of the finite difference grid. This updates `localgrid` and returns a local error estimate, `localmax`. Next, each task calls `global_maximum` to obtain a global maximum error, which is used to determine whether the simulation has converged. On a parallel computer, both the `finite_difference` and `global_maximum` routines must perform communication (to exchange the data required by the finite difference stencil and to compute the global maximum, respectively), but this activity is hidden from the rest of the program.

This example illustrates an important advantage of sequential composition and the SPMD model: the program executed by each process has a fairly straightforward sequential reading, and many sequential programming techniques can be used unchanged. For

example, the procedures `finite_difference` and `global_maximum` can be defined in separate grid and reduction modules, both of which can encapsulate internal data structures (and communication structures).

A second advantage of sequential composition is that if different modules use the same data distribution, no data movement (and hence no communication) is required at module interfaces. For example, the top-level structure of an SPMD climate modeling system could be as follows. Procedures from ocean and atmosphere modules are called repeatedly in an interleaved fashion, with data generated by the ocean module being passed to the atmosphere module and vice versa. Communication is required only within the two components (Foster, 1995).

```
initialize_ocn(ocn_grid)
initialize_atm(atm_grid)
 while (not done) do
      ocean(atm_grid, ocn_grid)
      atmosphere(ocn_grid, atm_grid, done)
enddo
```

Parallel Composition

Parallel composition can be viewed as a generalization of the SPMD programming model in which different parts of a computer execute different programs. (It can also be thought of as a special case of concurrent composition in which concurrently executing tasks are required to execute on disjoint sets of processors.) A parallel composition specifies which program components are to execute in which parts of the computer and how these components are to exchange data.

In principle, any program expressed as a parallel composition can be converted to a sequential composition that interleaves the execution of the various program components appropriately. However, the use of parallel composition can enhance scalability and locality. For example, if two program components (such as the atmosphere and ocean model considered in the preceding section) can execute concurrently, then mapping them to disjoint sets of processors increases scalability by providing additional opportunities for

parallel execution. If locality increases with granularity, then this parallel composition can also make more efficient use of cache, memory, and communication bandwidth than can a sequential composition of the same components. Parallel composition can also decrease total memory requirements by reducing the amount of code and data replicated on every processor (Foster, 1995).

Concurrent Composition

Concurrent composition is the most general form of composition that we consider. A concurrent composition specifies the program components that are to execute concurrently, producer/consumer relationships between components, and the mapping of components to processors. Components then execute in a data-driven manner, meaning that they can be executed if the data that they require from other components are available. These ideas should be familiar from the discussion of the task/channel programming model. In the terms of that model, a concurrent composition specifies a set of tasks, a set of channels connecting these tasks, and a mapping of tasks to processors (Foster, 1995).

Concurrent composition has both advantages and disadvantages relative to sequential and parallel composition. One important advantage is that it can facilitate information hiding and hence the development of modular programs. This is because the interfaces in a concurrent composition consist entirely of the channels connecting the various components. Internal implementation details concerning code, data structures, concurrency, and communication are hidden. Hence, program components can be designed and developed in isolation even when they need to execute on the same processors.

Concurrent composition can also simplify design by allowing decisions concerned with mapping and scheduling to be delayed or even avoided altogether. Because the semantics of a program specified by using concurrent composition are independent of how program components are mapped to processors, mapping decisions can be delayed until late in the design process, as recommended. Because the execution schedule is determined by the availability of data, execution order need not be specified explicitly by the programmer.

A disadvantage of concurrent composition in some environments is the cost of a data-driven execution model. While compilers and runtime systems can do much to reduce costs incurred when switching between tasks, these costs can be significant if task switches

occur frequently (Foster, 1995).

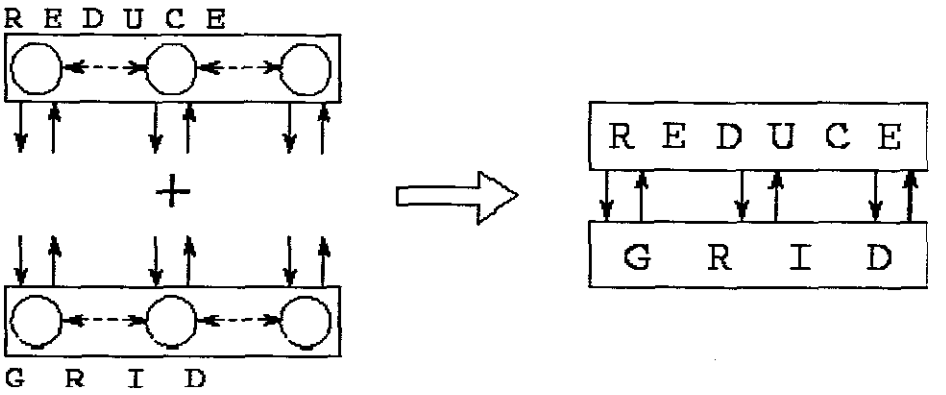


Figure 5.7: *A finite difference program can be structured as a concurrent composition of* `reduce` *and* `grid` *components. The first of these components is designed to perform reductions, while the second performs finite difference computation. An array of channels defines the interface between the two components, which encapsulate internal task and channel structures. The two components may execute on the same or different processors.*

5.3 The Distributed Computing Toolbox and the MATLAB Distributed Computing Engine

MATLAB

MATLAB is one of the most widely used tools in scientific and technical computing. It started in the 1970s as an interactive interface to EISPACK (Smith *et al.*, 1976) and LINPACK (Dongarra *et al.*, 1976), a set of eigenvalue and linear system solution routines. It has since grown to a feature-rich product utilizing modern numerical libraries such as ATLAS (Clint Whaley *et al.*, 2001) and FFTW (Frigo and Johnson, 1998) and with toolboxes in a number of application areas, for example, financial mathematics, neural networks, and control theory. It has a built-in interpreted language that is similar to Fortran 90, and the flexible matrix indexing makes it very suitable for programming matrix problems. Also, it provides hooks to Java classes and dynamically linked libraries, making integration with compiled code easy.

MATLAB gained popularity because of its user-friendliness. It has seen widespread use in

classrooms as a teaching tool. Its strong graphical capabilities makes it a good data analysis tool. Also, researchers have been known to build very complex systems using MATLAB scripts and toolboxes.

Why there should be a parallel MATLAB

Because of its roots in serial numerical libraries, MATLAB has always been a serial program. In 1995, C. Moler of Mathworks wrote a paper (Moler, 1995) stating Mathworks' intention not to develop a parallel MATLAB at that time. His arguments could be summarized as follows.

1) *Memory model:* Distributed memory was the dominant model for parallel computers, and for linear algebra applications, scatter/gather of the matrix took too long to make parallel computation worthwhile.

2) *Granularity:* For typical use, MATLAB spends most of its time in the parser, interpreter and graphics routines, where any parallelism is difficult to find. Also, to handle embarrassingly parallel applications, which only requires a collection of results at the end, MATLAB would require fundamental changes in its architecture.

3) *Business situation:* There were not enough customers with parallel computers to support the development.

It has been nine years since the article was written, and we have seen tremendous changes in the computing world. These changes have invalidated the arguments that there should not be a parallel MATLAB.

1) *Memory model:* As modern scientific and engineering problems grow in complexity, the computation time and memory requirements skyrocket. The increase in processor speed and the amount of memory that can fit in a single machine could not catch up with the pace of computation requirements. Very often, current scientific problems simply do not fit into the memory of a single machine, making parallel computation a necessity. This has always been true throughout the history of computing—problems just do not fit into memory. But we have hit the point where a lot of interesting and practical problems from different areas do not fit into memory of a single machine. This has made parallel computing a necessity.

2) *Granularity:* Over the past eight years simple parallel MATLAB projects, some consisting of only 2 m-files, have shown that multiple MATLAB instances running on a

parallel computer could be used to solve embarrassingly parallel problems, without any change to MATLAB itself. Also, increase in problem sizes and processor speed have reduced the portion of time spent in noncomputation related routines.

3) *Business situation:* The past few years have seen the proliferation of Beowulf clusters. Beowulf clusters are parallel computers made from commodity off-the-shelf (COTS) hardware. They often consist of workstations connected together with Ethernet or other common, nonproprietary interconnect. Researchers prefer Beowulf clusters over traditional supercomputers because Beowulfs are quite easy to set up and maintain and are cheap enough so that a researcher can get his or her "personal supercomputer." Also, inthe financial industry, large "compute farms" which are basically large clusters are widely used for various pricing and value at risk (VaR) simulations. However, often researchers in science wanting to use a parallel computer to solve problems are not experts in parallel programming. The dominant way of parallel programming, Message Passing Interface (MPI), is too low level and too error prone. MATLAB is well known forits user-friendliness. There is a huge potential market for a MATLAB that could be used to program parallel computers.

The Distributed Computing Toolbox and the MATLAB Distributed Computing Engine

The Distributed Computing Toolbox and the MATLAB Distributed Computing Engine are very popular for parallel processing now. The Distributed Computing Toolbox and the MATLAB Distributed Computing Engine give possibilities to coordinate and execute independent MATLAB operations simultaneously on a cluster of computers, speeding up execution of large MATLAB jobs (*Distributed Computing Toolbox For Use with MATLAB®*, 2005).

A *job* is some large operation that is needed to be performed in the MATLAB session. A job is broken down into segments called *tasks*. The user decides how best to divide the job into tasks. He could divide his job into identical tasks, but tasks do not have to be identical.

The MATLAB session in which the job and its tasks are defined is called the *client* session. Often, this is on the machine where the user programs MATLAB. The client uses the

Distributed Computing Toolbox to perform the definition of jobs and tasks. The MATLAB Distributed Computing Engine is the product that performs the execution of the job by evaluating each of its tasks and returning the result to the client session.

The *job manager* is the part of the engine that coordinates the execution of jobs and the evaluation of their tasks. The job manager distributes the tasks for evaluation to the engine's individual MATLAB sessions called *workers*. A schematic diagram of the client, job manger and workers is given on Fig 5.8.

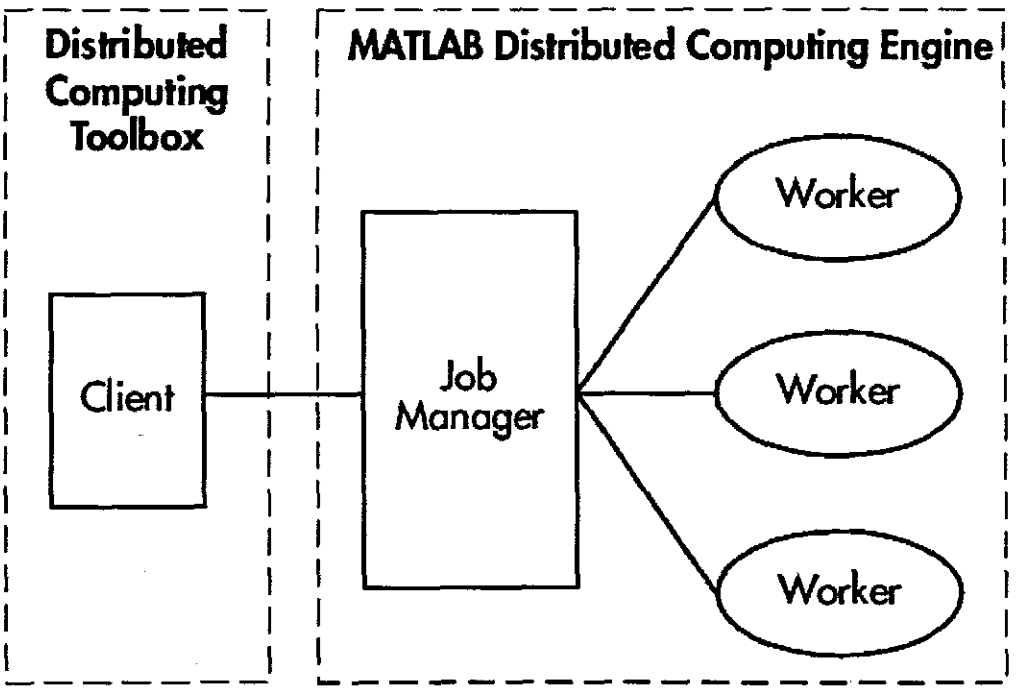


Figure 5.8 *Client , Job Manager and Worker*

Basic Distributed Computing Configuration

The following table summarizes the distributed computing terms introduced so far. The next section more fully explains these terms.

MATLAB Distributed Computing Terms

Name	Description
Job	The complete large-scale operation to perform in MATLAB, composed of a set of tasks.

Task	One segment of a job to be evaluated by a worker.
Client	The MATLAB session that defines a job using the Distributed Computing Toolbox.
Job manager	The part of the MATLAB Distributed Computing Engine that coordinates job execution, distributing tasks to individual workers for evaluation. This is represented in the client session by a job manager object.
Worker	The session of MATLAB in the MATLAB Distributed Computing Engine that evaluates tasks by executing the tasks' functions. This is represented in the client session by a worker object.

Table 5.1 The distributed computing terms

Job Managers, Workers, and Clients

The job manager can be run on any machine on the network. The job manager runs jobs in the order in which they are submitted, unless any jobs in its queue are promoted, demoted, canceled, or destroyed.

Each worker is given a task from the running job by the job manager, executes the task, returns the result to the job manager, and then is given another task. When all tasks for a running job have been assigned to workers, the job manager starts running the next job with the next available worker.

A MATLAB Distributed Computing Engine setup usually includes many workers that can all execute tasks simultaneously, speeding up execution of large MATLAB jobs. It is generally not important which worker executes a specific task. The workers evaluate tasks one at a time, returning the results to the job manager. The job manager then returns the results of all the tasks in the job to the client session.

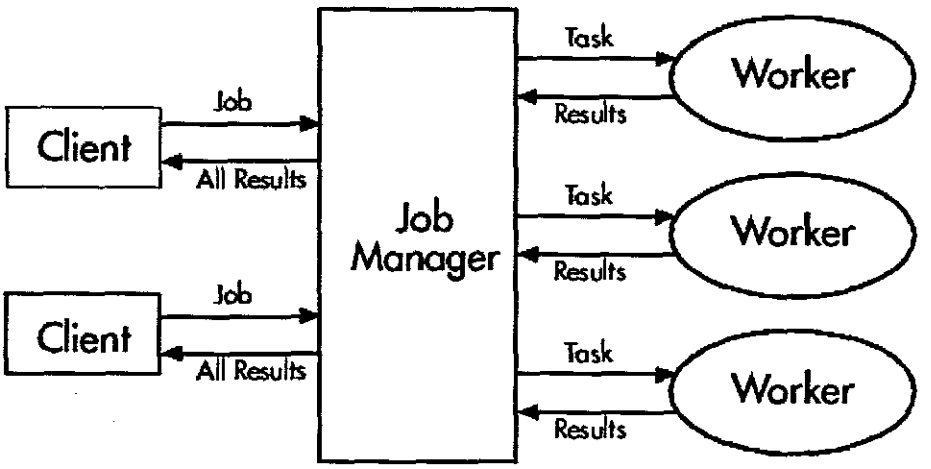


Figure 5.9 *Job and Task*

Interactions of Distributed Computing Sessions

A large network might include several job managers as well as several client sessions. Any client session can create, run, and access jobs on any job manager, but a worker session is registered with and dedicated to only one job manager at a time. The following figure shows a configuration with multiple job managers.

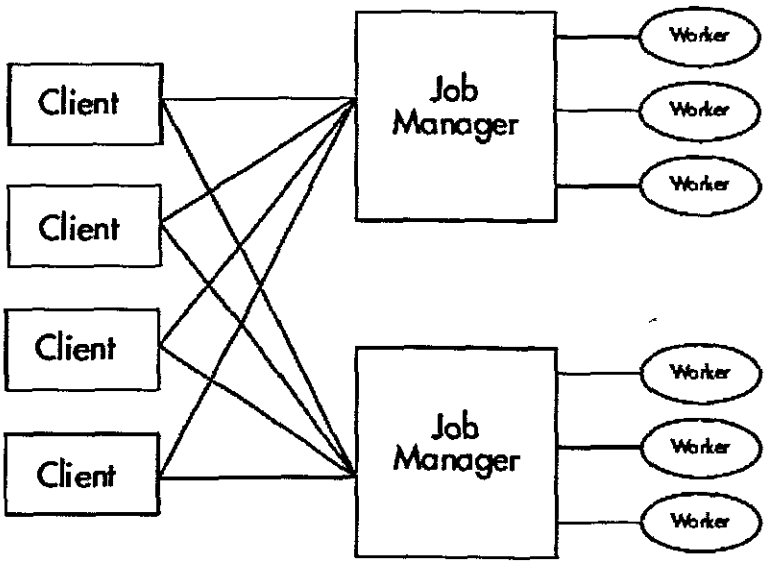


Figure 5.10 A configuration with multiple job managers

Configuration with Multiple Clients and Job Managers

Components on Mixed Platforms

The Distributed Computing Toolbox and MATLAB Distributed Computing Engine are supported on Windows, UNIX, and Macintosh platforms. Mixed platforms are supported, so that the clients, job managers, and workers do not have to be on the same platform.

The MATLAB Distributed Computing Engine Daemon

Every machine that hosts a worker or job manager session must also run the MATLAB Distributed Computing Engine (MDCE) Service. The MDCE daemon makes it possible for these processes on different machines to communicate with each other.

The MDCE daemon also recovers worker and job manager sessions when their host machines crash. If a worker or job manager machine crashes, when MDCE starts up again (usually configured to start at machine boot time), it automatically restarts the job manager and worker sessions to resume their sessions from before the system crash.

Components Represented in the Client

A client session communicates with the job manager by calling methods and configuring properties of a *job manager object*. Though not often necessary, the client session can also access information about a worker session through a *worker object*.

When a job is created in the client session, the job actually exists in the job manager. The client session has access to the job through a *job object*. Likewise, tasks that are defined for a job in the client session exist in the job manager, and they are accessed through *task objects*.

A typical Distributed Computing Toolbox client session includes the following steps. Details of each step appear in Creating and Running Jobs. A basic example follows in the next section.

1. **Find a Job Manager** -- The network may have one or more job managers available. The function used to find a job manager creates an object in the current MATLAB session to represent the job manager that will run the job.

2. **Create a Job** -- A job is created to hold a collection of tasks. The job exists on the job manager, but a job object in the local MATLAB session represents that job.

3. **Create Tasks** – While the job is in the pending state, you can create tasks can be created to add to the job. Each task of a job can be represented by a task object in the local MATLAB session.

4. **Submit a Job to the Job Queue for Execution** – When the job has all its tasks defined, it is submitted to the queue in the job manager. The job manager distributes the job's tasks to the worker sessions for evaluation. When all of the workers are completed with the job's tasks, the job manager moves the job to the finished state.

5. **Retrieve the Job's Results** – The resulting data from the evaluation of the job is available as a property value of each task object.

The Distributed Computing Toolbox and the MATLAB Distributed Computing Engine have the features:

1. Distributed execution of coarse-grained MATLAB algorithms and Simulink models on remote MATLAB sessions
2. Control of the distributed computing process via a function-based or an object-based interface
3. Distributed processing on both homogeneous and heterogeneous platforms
4. Support for synchronous and asynchronous operations
5. Access to single or multiple clusters by single or multiple users

5.4 Working with the Distributed Computing Toolbox and the MATLAB Distributed Computing Engine

The Distributed Computing Toolbox enables to run MATLAB or Simulink applications (jobs) on a computer cluster by dividing them into independent tasks. Each task evaluates a specified MATLAB function or Simulink model. A typical job might be divided into tasks that operate on unrelated data sets or individual sections of very large data sets, greatly speeding up data-intensive applications.

After the job is submitted for execution, the MATLAB Distributed Computing Engine executes each of its tasks. The engine consists of a job manager that coordinates the distribution of tasks and remote MATLAB sessions (workers) that execute the tasks. Once the workers complete their tasks, they send results back to the job manager, where the

client can access them using the Distributed Computing Toolbox (https://tagteamdbserver.mathworks.com/ttserverroot/Download/29492_91263v01_DCT_datasheet.pdf).

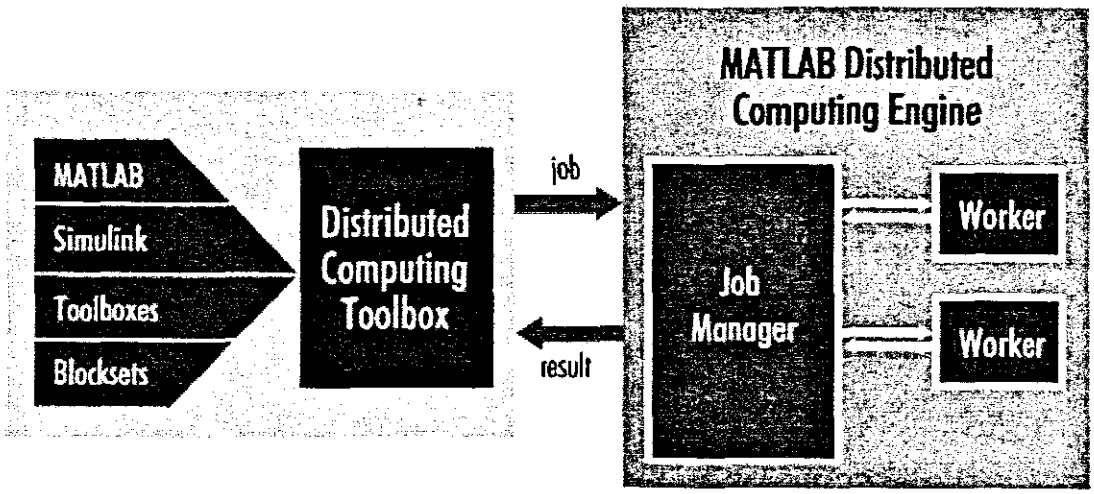


Figure 5.11: *The interaction between the client machine, where the Distributed Computing Toolbox is used to define jobs and tasks, and the MATLAB Distributed Computing Engine.*

Creating and Submitting Jobs with the Distributed Computing Toolbox

The Distributed Computing Toolbox makes it easy to define and submit jobs from the command line. The toolbox includes functions for defining jobs, dividing them into tasks, sending them to the MATLAB Distributed Computing Engine for execution, and retrieving the results. The complete process includes five steps:

Finding a job manager

Creating a job

Creating tasks

Submitting the job to the job queue

Retrieving results

Using the function-based interface, the client can go through the entire process with a single command. Alternatively, he can use the object-based interface to control each step.

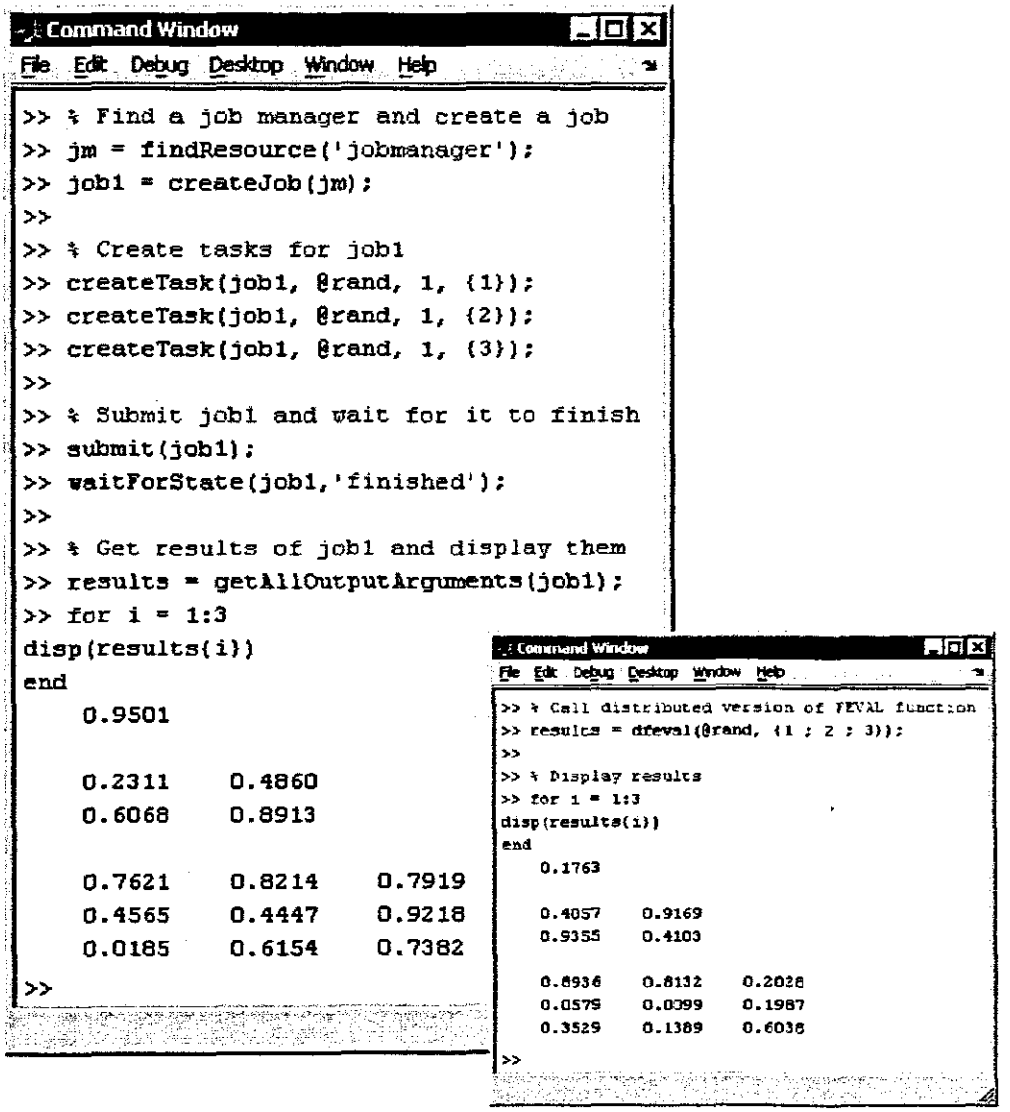


Figure 5.12: *Setting up and running a distributed application with the object-based interface (left) and the function-based interface (bottom). The disp function displays the results of each iteration of the for loop.*

Executing Jobs with the MATLAB Distributed Computing Engine

After the client defines and submits the job, the MATLAB Distributed Computing Engine workers evaluate each of the job's tasks and make the results available for retrieval by Distributed Computing Toolbox functions. The workers can execute algorithms that include any toolboxes or blocksets for which the client is licensed. In the computer cluster, no additional MathWorks product licenses are required beyond the MATLAB Distributed Computing Engine license.

The job manager in the MATLAB Distributed Computing Engine coordinates the execution of jobs and asynchronously distributes tasks to the workers. It can run on any machine on the network and can accept jobs from multiple users. The job manager runs jobs in the order in which they are submitted unless jobs are promoted, demoted, canceled, or destroyed. Once all tasks for a running job have been assigned to the workers, the job manager starts running the next job. Individual or multiple users can send jobs to single or multiple job managers.

5.5. Engine for parallel simulation

Each simulation tool exploits a *simulation engine* to model and execute the simulation model. Figure 5.13 shows the taxonomy for the simulation engine (Sulistio *et al.*, 2004). Simulations can be executed in serial or parallel modes. A serial or sequential simulation is executed using a single processor, while a parallel or distributed simulation is executed using multiple processors, located in PDSs (parallel and distributed systems). A serial simulation is restricted by limited memory and needs a longer execution time, compared with a parallel simulation. Therefore, there is an increasing interest in using parallel simulation for modeling complex, large-scale systems.

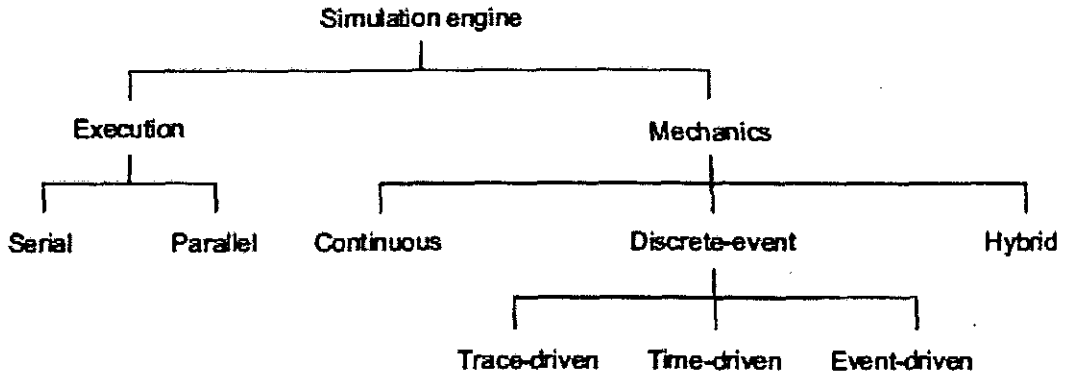


Figure 5.13 *Simulation engine taxonomy.*

Simulation models that are created for serial execution cannot be executed using parallel execution. In other words, the user can only design simulation models based on the type of execution mode supported by the simulation tool.

Most existing simulation tools use serial simulation. This is because simulation tools that use a parallel simulation engine are more difficult to implement. However, there is a growing trend to implement tools that exploit parallel simulation given the growing demand for faster simulation and shorter execution time. Some tools such as Parsec (Bagrodia *et al.*, 1998) and GloMoSim (Zeng *et al.*, 1998) support both serial and parallel simulation, which provides the flexibility for the user to use.

The simulation tool advances the simulation based on the mechanics defined in the simulation engine. In a continuous simulation, state changes occur continuously across time. In a discrete-event simulation (DES), state changes only occur at specific time intervals. A hybrid simulation comprises both continuous and discrete-event simulations. A DES adopts a queuing system where queues of events wait to be activated. A DES is further subdivided into a trace-driven, time-driven or event-driven simulation. A trace-driven DES proceeds by reading in a set of events that are collected independently from another environment and are suitable for modeling a system that has executed before in another environment. The user can trace and modify the inputs to observe and control the simulation. A time-driven DES advances by fixed time increments and is useful for modeling events that occur at regular time intervals. An event-driven DES advances by irregular time increments and is useful for modeling events that may occur at any time. An event-driven DES is more efficient than a time-driven DES since it does not step through regular time intervals when no event occurs. Most of the simulation tools surveyed in this paper use an event-driven DES because it is relevant for the context of simulating most large-scale PDSs and requires less time. However, hybrid simulation is needed by embedded system simulators such as Ptolemy II (Liu *et al.*, 2003).

5.6. Parallelization of the sequential algorithm for calculation of optimal control

The algorithm described in chapter 4 is easy parallelizable because it realizes a decomposition method which naturally decomposes the structure of the problem for optimization. The algorithm can be and is programmed in both-sequential and parallel way. The goal of parallelization is to obtain high performance and increased speed over the best sequential program that solves the same problem. This is to ensure efficient load

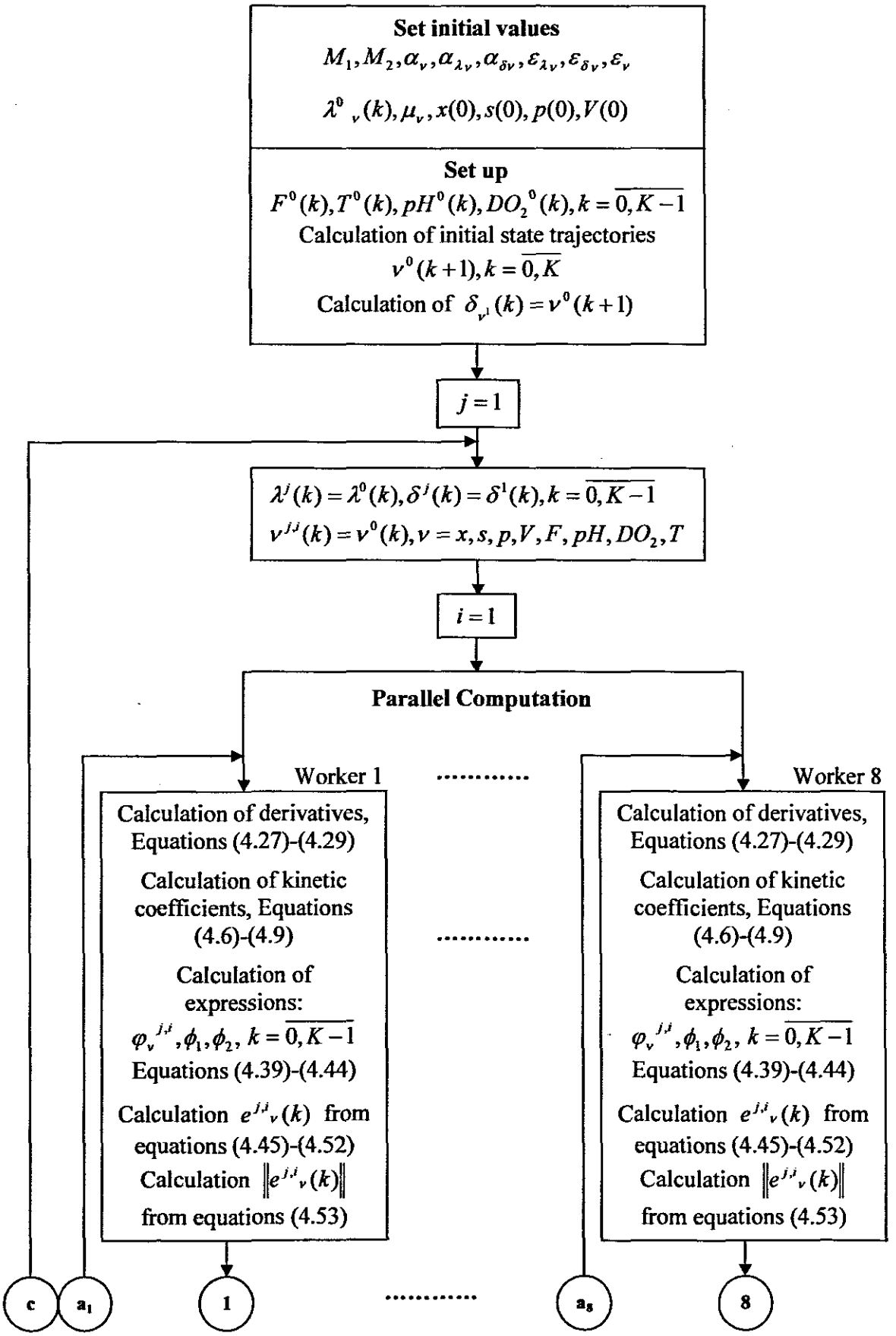
balancing among processors, reduction in communication overhead and synchronization. The parallelization is based on the following steps, contributing to achieve better performance:

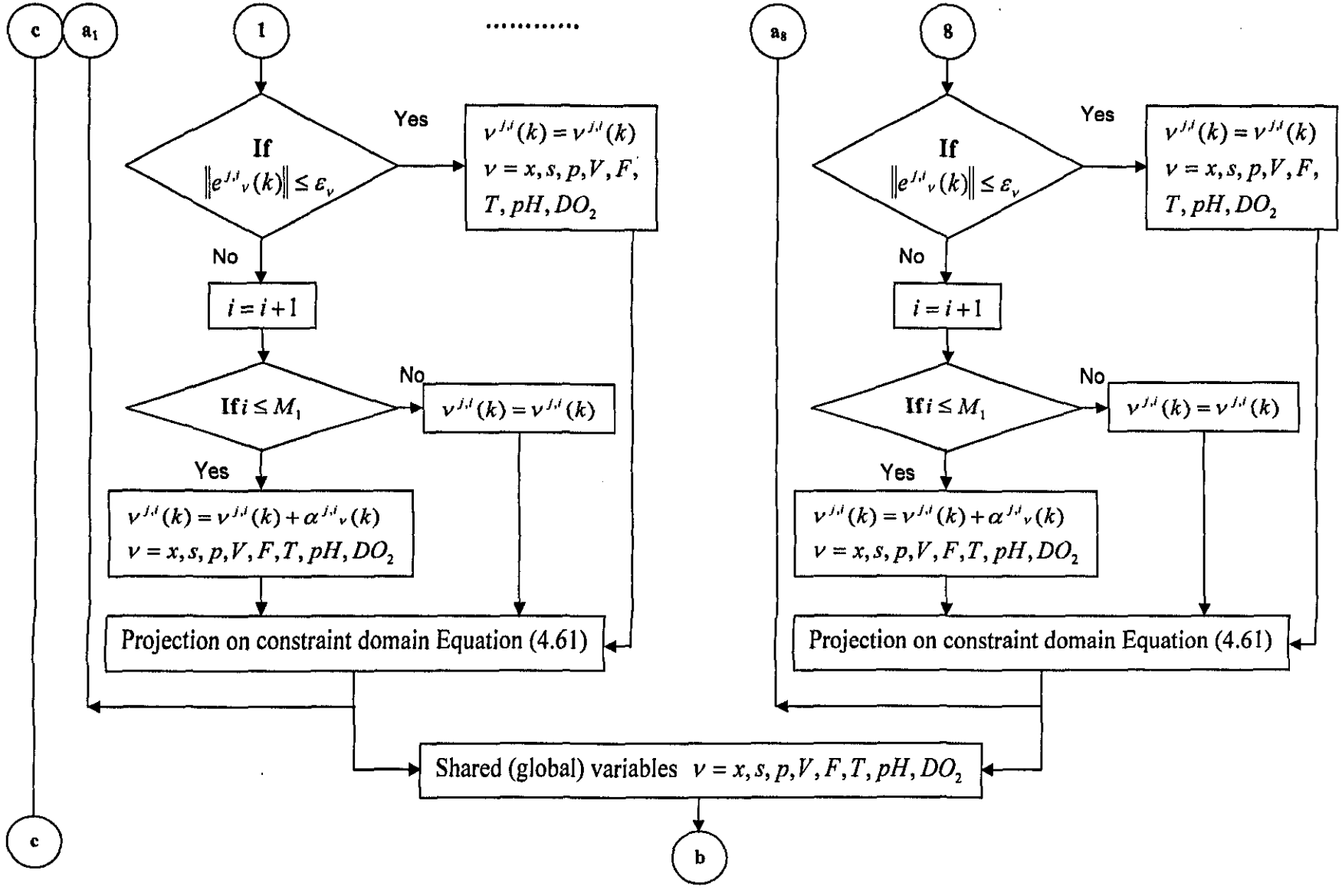
- Decomposing computation into tasks
- Assigning tasks to processes
- Orchestration: data accessing and establishing communication and synchronization among processes
- Mapping processes to processors for execution

A program in the parallel executable format is composed of a number of processes, each of which performs a subset of tasks in the program. Processes usually need to communicate and synchronize with one another.

Decomposition

The sequential algorithm Fig 4.2 is analyzed and examined to find the amount of parallelization inherent in the algorithm. In this case, this is the decomposition of the first level problem in sub-problems for every time moment. As the number of workers is 8, and the number of processes is 8, the time horizon is decomposed in 8 sub-horizons in such a way that every worker has fixed load size. To ensure better performance all processes are required to be kept busy as much of the time as possible through the decomposition. The parallel version of the sequential algorithm is given on Fig 5.14.


Set initial values
$M_1, M_2, \alpha_\nu, \alpha_{\lambda\nu}, \alpha_{\delta\nu}, \varepsilon_{\lambda\nu}, \varepsilon_{\delta\nu}, \varepsilon_\nu$
$\lambda^0{}_\nu(k), \mu_\nu, x(0), s(0), p(0), V(0)$
Set up
$F^0(k), T^0(k), pH^0(k), DO_2{}^0(k), k = \overline{0, K-1}$
Calculation of initial state trajectories
$\nu^0(k+1), k = \overline{0, K}$
Calculation of $\delta_{\nu^1}(k) = \nu^0(k+1)$
$j = 1$
$\lambda^j(k) = \lambda^0(k), \delta^j(k) = \delta^1(k), k = \overline{0, K-1}$
$\nu^{j,j}(k) = \nu^0(k), \nu = x, s, p, V, F, pH, DO_2, T$
$i = 1$
Parallel Computation
Worker 1
Calculation of derivatives, Equations (4.27)-(4.29)
Calculation of kinetic coefficients, Equations (4.6)-(4.9)
Calculation of expressions:
$\varphi_\nu{}^{j,i}, \phi_1, \phi_2, k = \overline{0, K-1}$
Equations (4.39)-(4.44)
Calculation $e^{j,i}{}_\nu(k)$ from equations (4.45)-(4.52)
Calculation $\|e^{j,i}{}_\nu(k)\|$
from equations (4.53)
Worker 8
Calculation of derivatives, Equations (4.27)-(4.29)
Calculation of kinetic coefficients, Equations (4.6)-(4.9)
Calculation of expressions:
$\varphi_\nu{}^{j,i}, \phi_1, \phi_2, k = \overline{0, K-1}$
Equations (4.39)-(4.44)
Calculation $e^{j,i}{}_\nu(k)$ from equations (4.45)-(4.52)
Calculation $\|e^{j,i}{}_\nu(k)\|$
from equations (4.53)
c
a1
1
a8
8


c
a1
1
.....
a8
8
If
$\|e^{j,i}{}_v(k)\| \le \varepsilon_v$
Yes
$v^{j,i}(k) = v^{j,i}(k)$
$v = x, s, p, V, F, T, pH, DO_2$
No
$i = i+1$
If $i \le M_1$
No
$v^{j,i}(k) = v^{j,i}(k)$
Yes
$v^{j,i}(k) = v^{j,i}(k) + \alpha^{j,i}{}_v(k)$
$v = x, s, p, V, F, T, pH, DO_2$
Projection on constraint domain Equation (4.61)
Shared (global) variables $v = x, s, p, V, F, T, pH, DO_2$
c
b

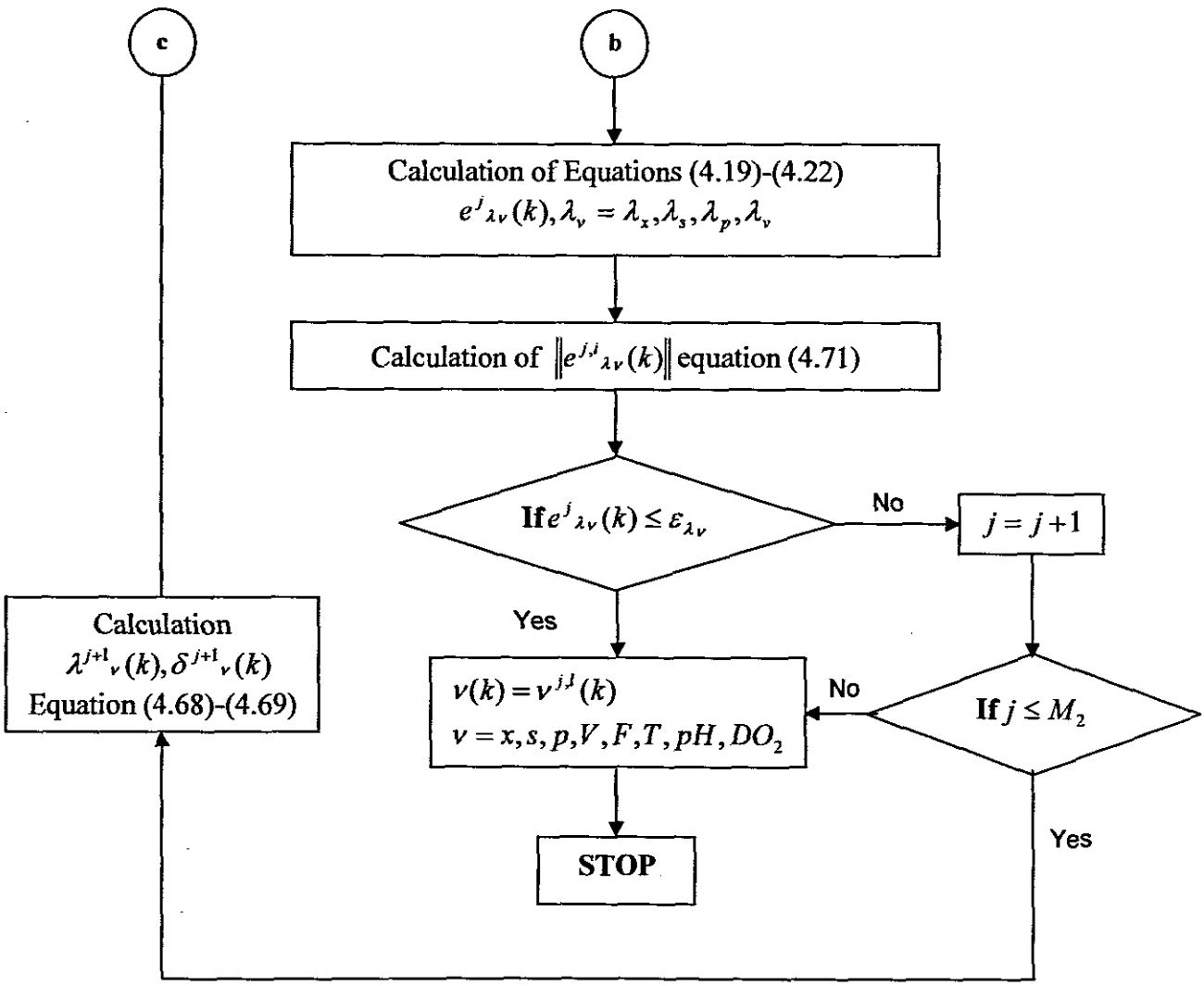


Fig 5.14 Parallelization of the computational algorithm

Assigning task to processes

In this step tasks are allocated among the processes with the objectives of

- Balancing the workload. This is referred as load balancing.
- Reducing the inter-process communication and runtime overhead.

Load balancing includes data access, computation and communication. Reducing inter-worker communication is very important for achieving better performance. In the considered case there is not such type of communication. Allocation of tasks is static one as the task are allocated to workers by analyzing the algorithm before the execution

has started and the allocation does not change during program execution. The allocation of tasks is:

First worker ------ time horizon for 6 steps, k = $\overline{1,6}$

Second worker ------ time horizon for 6 steps, k = $\overline{7,12}$

Third worker ------ time horizon for 6 steps, k = $\overline{13,18}$

Forth worker ------ time horizon for 6 steps, k = $\overline{19,24}$

Fifth worker ------ time horizon for 6 steps, k = $\overline{25,30}$

Sixth worker ------ time horizon for 6 steps, k = $\overline{31,36}$

Seventh worker ------ time horizon for 6 steps, k = $\overline{37,42}$

Eight worker ------ time horizon for 6 steps, k = $\overline{43,48}$

Worker process NO.	Length of the sub-horizon	Position in the common horizon
1	6	k = $\overline{1,6}$
2	6	k = $\overline{7,12}$
3	6	k = $\overline{13,18}$
4	6	k = $\overline{19,24}$
5	6	k = $\overline{25,30}$
6	6	k = $\overline{31,36}$
7	6	k = $\overline{37,42}$
8	6	k = $\overline{43,48}$

Table 5.2 Decomposition of the optimization horizon between the workers

The last point for k = 49 is computed in the worker with index1.

Orchestration

Accessing data, exchanging data among two workers and ensuring appropriate synchronization belong to this step. Programming plays a vital role in performing all these steps. The main factors considered are organizing data structures, scheduling the tasks assigned to the workers, whether to communicate explicitly or implicitly in order

to reduce the cost of communication and synchronization, preserving locality of data reference and reducing the overhead of parallelism management. The functions of the Matlab Distributed toolbox are used to achieve the goals of this step.

Mapping the processes to workers for execution

This step involves mapping the processes to processors for execution. Here it is performed manually in such a way that every process has one worker and the processes do not migrate from one worker to another during execution. The mapping is done with the help of the Matlab Job manager and the special functions of Matlab Distributed Computing toolbox for parallel tasks.

Programming model

Used parallel programming model is the synchronous one. It allows parallelism only within given steps of solution. A program specifies a sequence of such steps, with parallelism facilitated by simultaneous execution of each step in many processors. This approach is often associated with on Single Input Multiple Data (SIMD) execution model. In such a model only a single sequence of instructions is present, but each instruction may be simultaneously executed in an arbitrary set of processors, with each processor operating upon its own data.

Synchronous model allows the problem to be decomposed into a series of small steps that may be executed in parallel.

The process of parallel computing is based on the study of data dependencies. The presence of dependence between two computations implies that they cannot be performed in parallel. The parallel processes communicate and synchronize by reading and writing shared variables. The processes in the program communicate and synchronize with each other when they send their results from the calculation and when they receive the new values of the coordinating variables. The existing two methods of synchronization are applied — synchronization for precedence and synchronization for mutual exclusion.

The method of synchronization for precedence guarantees that one event does not begin until another event has finished. The semaphores are used for these purposes. The semaphores are locking mechanisms used for fill locking. One can use them to control access to files, shared memory and I/O devices. In the considered case the coordinating

process has to wait for completion of the processes in the workers. The calculation in each process can be with different length depending on the convergence of the gradient procedures.

The method of synchronization for mutual exclusion guarantees that only one process can access the critical section where the data are shared and must be manipulated. The mutual exclusion mechanism is used for this purpose. This means if one process is executing in its critical section, other processes will be excluded from doing the same thing. The worker with index1 is used for this purpose in Matlab.

Inter-process communication uses an explicit communication mechanism as message passing given by the function *broadcast*. The processes also share all global variables which give plenty of opportunities for synchronization.

The used inter-process communication mechanisms are global variables, shared memory and messages passing.

The parallel version of the Matlab programme is described in chapter 6.

5.7. Conclusion

The concepts and evolution of parallel and distributed computing, the distributed computing toolbox and MATLAB distributed computing Engine, the building of the distributed computing programmes are all described in this chapter. The inputs, outputs, functions and results of the programmes for parallel optimal control calculation in MATLAB Distributed Computing Toolbox will be given in the next chapter.

Chapter 6

Description and calculation results from the Matlab programs for modeling and optimal control

6. Introduction

This chapter describes the programmes for modeling and sequential and parallel optimization in MATLAB using Matlab/Simulink and its distributed computing engine and toolbox.

6.1. Correspondence between mathematical and programs notations

The correspondence between the mathematical notations and a program notations is given in Table 6.1.

Name	Notation in the text	Notation in Matlab
State variable - Biomass - Substrate - Product - volume	 x s p v	 x s p v
Control variable - Temperature - PH - DO_2 - Flow rate	 T pH DO_2 F	 T pH DO2 F
Steps of the gradient procedures for state variables	α_x α_s α_p α_v $\alpha_{\lambda x}$ $\alpha_{\delta x}$	ax as ap av alx adx
Conjugate variable	λ_x λ_s λ_p λ_v	lx ls lp lv
Interconnections in time domain	δ_x	dx

	δ_s δ_p δ_v	ds dp dv
Coefficients	$a_1, a_2, a_3, a_4, a_5, a_6, a_7$ $b_1, b_2, b_3, b_4, b_5, b_6, b_7$ $c_1, c_2, c_3, c_4, c_5, c_6, c_7$ $d_1, d_2, d_3, d_4, d_5, d_6, d_7$ K_d K_s m n $Y_{x/s}$ $Y_{p/x}$ s_i p^* Δt μ_{max}	a1, a2, a3, a4, a5, a6, a7 b1, b2, b3, b4, b5, b6, b7 c1, c2, c3, c4, c5, c6, c7 d1, d2, d3, d4, d5, d6, d7 Kd Ks m n Yxs Ypx si pl dt mmax
Initial values of state variables	x(0) s(0) p(0) v(0)	x0 s0 p0 v0
Penalty coefficents	μ_x μ_s μ_p μ_v	mx ms mp mv
Min and max of state variables	x_{max} x_{min} s_{max} s_{min} p_{max} p_{min} v_{max} v_{min}	xmax xmin smax smin pmax pmin vmax vmin
Min and max of control variables	T_{max} T_{min} pH_{max} pH_{min} DO_{2max} DO_{2min} F_{max} F_{min}	Tmax Tmin pHmax pHmin DO2max DO2min Fmax Fmin
Derivatives of the model parameters	$\frac{\partial \mu_{max}}{\partial T} = a_2 + 2a_3T(k)$ $\frac{\partial K_s}{\partial T} = b_2 + 2b_3T(k)$	dmmaxdT(k) = a2+2*a3*T(k); dKsdT(k)=b2+2*b3*T(k);

	$\frac{\partial Y_{x/s}}{\partial T} = c_2 + 2c_3T(k)$	dYxsdT(k)=c2+2*c3*T(k);
	$\frac{\partial Y_{p/s}}{\partial T} = d_2 + 2d_3T(k)$	dYpxdT(k)=d2+2*d3*T(k);
	$\frac{\partial \mu_{max}}{\partial pH} = a_4 + 2a_5pH(k)$	dmmaxdpH(k)=a4+2*a5*pH(k);
	$\frac{\partial K_s}{\partial pH} = b_4 + 2b_5pH(k)$	dKsdpH(k)=b4+2*b5*pH(k);
	$\frac{\partial Y_{x/s}}{\partial pH} = c_4 + 2c_5pH(k)$	dYxsdpH(k)=c4+2*c5*pH(k);
	$\frac{\partial Y_{p/x}}{\partial pH} = d_4 + 2d_5pH(k)$	dYpxdpH(k)=d4+2*d5*pH(k);
	$\frac{\partial \mu_{max}}{\partial DO_2} = a_6 + 2a_7DO_2$	dmmaxdDO2(k)=a6+2*a7*DO2(k);
	$\frac{\partial K_s}{\partial DO_2} = b_6 + 2b_7DO_2(k)$	dKsdDO2(k) =b6+2*b7*DO2(k);
	$\frac{\partial Y_{x/s}}{\partial DO_2} = c_6 + 2c_7DO_2(k)$	dYxsdDO2(k) = c6+2*c7*DO2(k);
	$\frac{\partial Y_{p/x}}{\partial DO_2} = d_6 + 2d_7DO_2(k)$	dYpxdDO2(k) = d6+2*d7*DO2(k);
Error	$e_{\lambda x}(k) = \partial x(k) - x(k+1)$	elx(k) = dx(k)-x(k+1)
	$e_{\lambda s}(k) = \partial s(k) - s(k+1)$	els(k) = ds(k)-s(k+1)
	$e_{\lambda p}(k) = \partial p(k) - p(k+1)$	elp(k) = dp(k)-p(k+1)
	$e_{\lambda v}(k) = \partial v(k) - v(k+1)$	elv(k) = dv(k)-v(k+1)
	$e_{\delta x} = \mu_x(k)e_{\lambda x}{}^j(k)$	edx(k)=mx*elx(k)
	$e_{\delta s} = \mu_s(k)e_{\lambda s}{}^j(k)$	eds(k)=ms*els(k)
	$e_{\delta p} = \mu_p(k)e_{\lambda p}{}^j(k)$	edp(k)=mp*elp(k)
	$e_{\delta v} = \mu_v(k)e_{\lambda v}{}^j(k)$	edv(k)=mv*elv(k)
	$e_{ex} = \lambda_x{}^j(k) - \mu_x(k)e_{\lambda x}{}^j(k)$	eex(k)=lx(k)-mx*elx(k)
	$e_{es} = \lambda_s{}^j(k) - \mu_s(k)e_{\lambda s}{}^j(k)$	ees(k)=ls(k)-ms*els(k)
	$e_{ep} = \lambda_p{}^j(k) - \mu_p(k)e_{\lambda p}{}^j(k)$	eep(k)=lp(k)-mp*elp(k)
	$e_{ev} = \lambda_v{}^j(k) - \mu_v(k)e_{\lambda v}{}^j(k)$	eev(k)=lv(k)-mv*elv(k)
Formulas	$\Delta t\mu_{max}\left[1 - \frac{p(k)}{p^*}\right]^n = \varphi_f(k)$	ff(k)=dt*mmax(k) *((1-pk/pl)^0.52)
	$\Delta t\mu_{max}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{s(k)}{K_s + s(k)} = \varphi_x(k)$	fx(k)=ff(k)*s(k)/(Ks(k) + s(k))

	$\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{[K_s+s(k)]^2}=\varphi_s(k)$	fs(k)=ff(k)*Ks(k) * x(k)/((Ks(k) + s(k))^2)
	$\frac{\Delta t\mu_{\max}Y_{p/x}}{p^*}\left[1-\frac{p(k)}{p^*}\right]^{n-1} \frac{x(k)s(k)}{K_s+s(k)}=\varphi_p$	fp(k)=dt*mmax(k)*Ypx(k)*((1-p(k)/pl)^(n-1))*x(k)*s(k)/(pl*(Ks(k) + s(k)))
	$\Delta t\frac{F(k)}{V^2(k)}=\varphi_v(k)$	fv(k)=dt*F(k)/(v(k)^2)
	$\Delta t\left[1-\frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s+s(k)}=\varphi_1(k)$	f1(k)=ff(k)*x(k)*s(k)/(Ks(k) + s(k))
	$\Delta t\mu_{\max}\left[1-\frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{[K_s+s(k)]^2}=\varphi_2(k)$	f2(k)=f1(k)/(Ks(k) + s(k))

Table 6.1 Correspondence between mathematical notations and Matlab notations

6.2. Description of the Matlab sequential program for optimal control

The program optconfer.m is developed to implement the decomposition method in a sequential way. The program calculates the optimal trajectories of the state and control variables.

6.2.1. Input parameters and variables

It is given in Appendix A.

global parameter: T, pH, DO_2, F, k_d, m, p^*, n, s_i.

Set Initial values

M1 = 5; M2 = 5; Max number of iterations of the first and second levels

K = 48; Number of steps in optimization horizon

$\alpha_x = 0.001$; $\alpha_s = 0.001$; $\alpha_p = 0.001$; $\alpha_v = 0.001$; Steps of the gradient procedures for calculation of the State variables

$\alpha_T = 0.05$; $\alpha_F = 0.05$; $\alpha_{pH} = 0.05$; $\alpha_{DO_2} = 0.05$; Steps of the gradient procedures for calculation of the Control variables

$\alpha_{\lambda x} = 0.001$; $\alpha_{\lambda s} = 0.001$; $\alpha_{\lambda p} = 0.001$; $\alpha_{\lambda v} = 0.001$; Step of the gradient procedures for calculation of the Conjugate variables

$\alpha_{\delta x}$ = 0.001; $\alpha_{\delta s}$ = 0.001; $\alpha_{\delta p}$ = 0.001; $\alpha_{\delta v}$ = 0.001; Steps of the gradient procedures for calculation of the Interconnections in time

ε_1 =0.001

ε_2 =0.001 tolerances for stop of the calculation of the first and second levels

ε_x = 0.01; ε_s = 0.01; ε_p = 0.01; ε_v = 0.01; State error variables

ε_T = 0.01; ε_T = 0.01; ε_{pH} = 0.01; ε_{DO_2} = 0.01; Control error variables

Initial values of the penalty coefficients

μ_x=0.0010

μ_s=0.0010

μ_p=0.0010

μ_v=0.0010

Initial trajectories for the conjugate variables for k = 1:K

λ_x (k)=0.001;

λ_s (k)=0.005;

λ_p (k)=0.002;

λ_v (k)=0.003;

Initial trajectories of the control variables for k = 1:K

T (k) = 16;

pH (k) = 4.8;

DO2 (k) = 37;

F (k) = 0.;

Model co-efficients

a_1 = 0.0004; a_2 = 0.0001; a_3 = 0.0002; a_4 = 0.0001; a_5 = 0.0001; a_6 = 0.00002; a_7 = 0.00002;

b_1 = 4; b_2 = 0.5; b_3 = 0.13; b_4 = 0.13; b_5 = 0.013; b_6 = 0.014; b_7 = 0.013;

$c_1 = 0.0002$; $c_2 = 0.0001$; $c_3 = 0.000021$; $c_4 = 0.00021$; $c_5 = 0.000211$; $c_6 = 0.0000211$; $c_7 = 0.0002$;

$d_1 = 0.0360$; $d_2 = 0.008$; $d_3 = 0.003$; $d_4 = 0.005$; $d_5 = 0.005$; $d_6 = 0.001$; $d_7 = 0.0002$;

par = [a_1 a_2 a_3 a_4 a_5 a_6 a_7 b_1 b_2 b_3 b_4 b_5 b_6 b_7 c_1 c_2 c_3 c_4 c_5 c_6 c_7 d_1 d_2 d_3 d_4 d_5 d_6 d_7];

$k_d = 0.0008$;

$p^* = 50$;

$m = 0.11$;

$n = 0.52$;

$s_i = 266.6$;

$\delta_t = 2.5$; %Delta t

Initial values of state variables

$x_0 = 1.83$

$s_0 = 150.0$

$p_0 = 2.1$

$v_0 = 2.1$

y_0 = [x0 s0 p0 v0]'

Constraints of state and control variables

For control variables

$T_{min} = 10$;

$pH_{min} = 4$;

$DO_{2min} = 10$;

$F_{min} = 0$;

$T_{max} = 20$;

$pH_{max} = 7$;

$DO_{2max} = 60$;

$F_{max} = 5.0$;

For State variables

$v_{min} = 2$;

$v_{max} = 4$;

$p_{min} = 2.0$;

$p_{max} = 70$;

$s_{min} = 20$;

$s_{max} = 300$;

$x_{min} = 0.83$;

$x_{max} = 55$;

6.2.2 Output variables

X ------ Optimal trajectory of state variables

P ------ Optimal trajectory of product concentrations

S ------ Optimal trajectory of substrate concentration

V ------ Optimal trajectory of volume

T ------ Optimal trajectory of temperature

PH ------ Optimal trajectory of pH

DO_2 ------ Optimal trajectory of dissolved oxygen

F ------ Optimal trajectory of flow rate

The program is given in Appendix A.

6.3. Program function model_func1.m for calculation of the trajectories of the state variables derivatives

The function model_func1 is used to calculate the trajectories of the derivatives of the state space vector, according to the Equations (4.2 – 4.9). This program is called by the function ode15s for integration of the model equations.

The input variables are

--- *t* --- the time at moment *t* from the considered time interval

--- *y* --- initial, or value of the state vector in the moment *t*

The output variables are the trajectories of the state space vector derivatives --- *ydot*.

The program is given in Appendix A.

6.4. Program in Simulink for process simulation

The program Opt_Model.mdl is developed to simulate the fermentation process model (4.1-4.71). It is given in Appendix B.

6.4.1. Input

pH, T, DO_2, F are inputs in the program of Simulink. The functions *From workspace* are used in the program to introduce time and input variables from Matlab workspace. The block's Data parameters are [t1,T1], [t1,pH1], [t1,O1], [t1,FF]. The Sample times are 0.

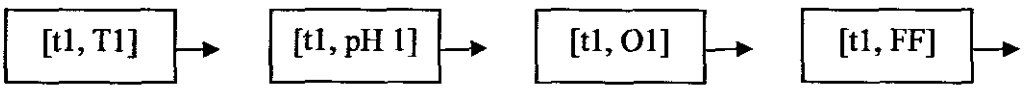


Fig 6.1 From workspace

6.4.2. Output

x, s, p, v are outputs in the program of Simulink. The functions *To workspace* are used in the program to send the trajectories of these variables to Matlab workspace. The Save format parameters are Array. The Limit data points to the last parameters are inf. The Decimation parameters are 1. The Sample time parameters are -1.

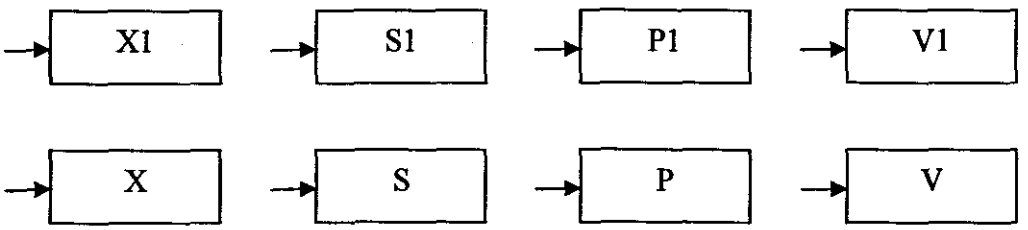


Fig 6.2 To workspace

6.4.3. Subsystem blocks

The kinetic coefficients μ_{max}, $Y_{x/s}$, $Y_{p/x}$, K_s as functions of the control variables T, pH, DO_2, F (4.23 - 4.26) are calculated in the program using *function* block. μ_{max}, $Y_{x/s}$, $Y_{p/x}$, K_s are described by Subsystems. μ_{max} calculation is given as an example in Fig 6.3.

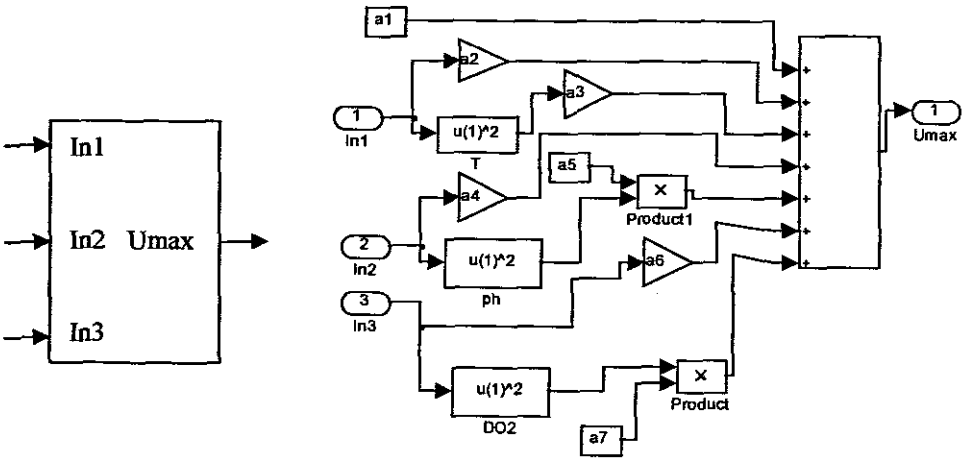


Fig 6.3 Subsystem and its contents

6.4.4. Function blocks

There are some Function blocks in the program: Fcn, Unit Delay, Mux, Sum and Product. The Expressions are (u[1]*u[2]*u[3])/u[4], u[1]/u[2], u[1]*kd*u[2], u[1]*m*u[2] and the Sample time are -1 in fcn, fcn2, fcn6.

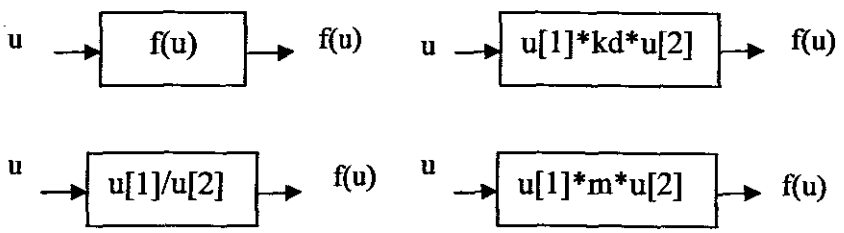


Fig 6.4 Fcn blocks for mathematical calculations

The discrete derivatives for the state variables x(k), s(k), p(k), v(k) are described by the Unit Delay block. The Initial conditions are x0, s0, p0, v0 and the Sample time are 1.25. Using x(k) as an example in Fig 6.5.

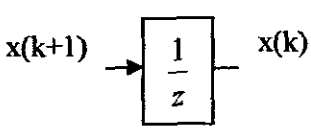


Fig 6.5 Unit Delay blocks

The Mux block combines its inputs into a single vector output. An input can be a scalar, vector, or matrix signal. Depending on its inputs, the output of a Mux block is a vector or a composite signal. The Mux block's Number of Inputs are 2, 3 and 4.

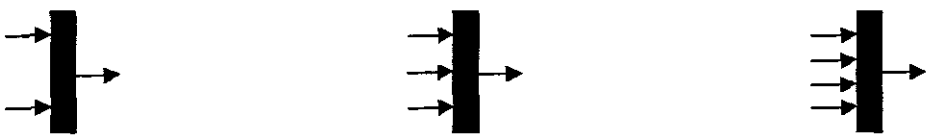


Fig 6.6 Mux block

The Sum block performs addition or subtraction on its inputs. This block can add or subtract scalar, vector, or matrix inputs. It can also sum the elements of a single input vector. The list of signs are |++--, |++-, |+--+ and |++ in this program. The Sample time are -1.

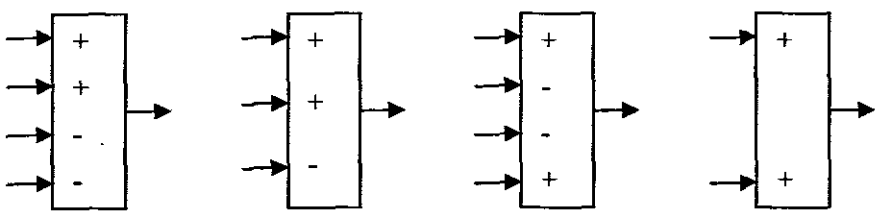


Fig 6.7 Sum block

The Product block performs multiplication or division of its inputs. The Number of inputs are 2, the Multiplications are Element-wise (.*) and the Sample time are -1 in the program.

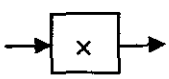


Fig 6.8 Product block

6.4.5. Script file for the input and output parameters

The script file YM_var.m is used to introduce the model parameters and control trajectories needed for the simulation in Simulink. It is given in Appendix B.

6.5. Description of the Matlab parallel program for optimal control

The program optconfer2.m realizes the sequential program for optimal control calculation using special functions and commands from the Matlab distributed computing toolbox.

6.5.1. Inputs

These inputs are the same with the inputs in the sequential program.

6.5.2. Outputs

These outputs are the same with the outputs in the sequential program.

6.5.3. Transformation of the inputs to implement parallel computing

As the parallel computing is based on shared resources and memory the global definition of the variables is used by the command:

global par T pH DO2 F kd m pl n si K M1 eps1 mu l x s p v el conmin conmax statemin statemax astate acon dt y0 x1 s1 p1 v1 T1 pH1 DO21 F1.

The steps of the gradient procedures for calculation of the state control and conjugate state variables and the values of the penalty coefficients are grouped in vectors.

astate=[ax as ap av]';

acon=[aT aF apH aDO2]';

al=[alx als alp alv]';

mu=[mx ms mp mv]';

The initial trajectories for the conjugate variables are grouped in a vector also.

l=[lx ls lp lv]';

The same is done for the min and max values of the state and control variables:

conmin=[Tmin pHmin DO2min Fmin]';

conmax=[Tmax pHmax DO2max Fmax]';

statemin=[xmin smin pmin vmin]';

statemax=[xmax smax pmax vmax]';

and for the errors $e_{\lambda v}(k), k = \overline{1,k}$.

el=[elx; els; elp; elv];

These vectors are global, the same ones for the main program given by a script files optconfer1(2).m and for the parallel subprogram given by function file subprog1(2).m.

Two variants of the program are developed with the help of the sub-program subprog1.m is used to organize the computations in the sequential way in the script file optconfer1.m. In the second case the subprogram is used to organize parallel computation in the cluster. In this way the user has three steps in the calculation

- Sequential one
- Sequential using the programs for parallel computation
- Parallel using the programs for parallel computation

These three steps of organizing of the calculation help to overcome many difficulties if the program is written directly for parallel computation.

6.5.4. Organizing of the parallel computation

The start of the coordinating procedure described in chapter4 is used for parallelization of the calculation of the state and control trajectories for 1/8 parts of the optimization interval.

```
j=1
while j <= M2;      Second level iterations
```

The functions for creation of jobmanager, job and task are called

```
jm=findResource('scheduler','type','jobmanager',...
    'name', 'Myjobmanager1','LookupURL','TzonevaRa')
pjob=createParallelJob(jm)
set(pjob,'MinimumNumberOfWorkers',8)
set(pjob,'MaximumNumberOfWorkers',8)
```

The function subprog2.m is on the client path, but it has to be available to the workers.One way to do this is to use the FileDependencies property of the pjob.

```
set(pjob,'FileDepandencies',{'subprog2.m'})
```

Starting parallel computing is done by creating a task which is the same for every worker. The task has only one output with 8 output arguments and not input arguments.

```
task(j)=createTask(pjob, @subprob,8,{})
submit(pjob)
```

The function subprog2.m is calculating the first level problem for 1/8 of the optimization interval. When the calculation completes, the output results are in the global variables x1, s1, p1, v1, T1, pH1, DO21, F1, transformed to the variables x, s, p, v, T, pH, DO2, F by the worker with index1

```
waitForState(pjob,'finished')
```

The results are used for calculation of the improved values of the coordinating variables, which again are used for calculation of state and control variables until the convergence of the coordination process is achieved.

6.5.5. Function for parallel calculation

The part of the program implemented in a parallel way is described by the function file subprog2.m.

```
function [x,s,p,v,T,pH,DO2,F]=subprog2
```

The same global parameters are declared.

```
global par T pH DO2 F kd m pl n si K M1 eps1 mu l x s p v el conmin conmax statemin
statemax astate acon dt y0 x1 s1 p1 v1 T1 pH1 DO21 F1.
```

The subprog2.m starts with calculation of the start and stop points of the optimization horizon for every worker.

```
dd=K/numlabs
a=(labindex-1)*dd+1
b=labindex*dd
```

Decomposition in time domain inside every subinterval is done

```
for k=a:b
    i=1
```

The function determines the first level iterations number.

```
    while i <= M1
```

Calculation of the expressions lv(k), k=0,K-1, v = x, s, p, v is followed by calculation of the errors and the values of the state and control variables. The iterations continue

until the state and control gradients are less than the error eps1 or until the number of iterations becomes equal to M1. The program has to reach this point for every worker. The function *labBarrier* is used for this purpose:

labBarrier

The Distributed Computing function *labBarrier* is used to synchronize the parallel calculations and to ensure that all workers have done the necessary calculations. Then the worker with labindex equal to 1 is forming the whole trajectory of state and control variables on the tasks of their global definition in the following way.

```
if labindex==1
   x2=reshape(x1,1,prod(size(x1)));
    s2=reshape(s1,1,prod(size(s1)));
    p2=reshape(p1,1,prod(size(p1)));
    v2=reshape(v1,1,prod(size(v1)));
    T2=reshape(T1,1,prod(size(T1)))
    pH2=reshape(pH1,1,prod(size(pH1)));
    DO22=reshape(DO21,1,prod(size(DO21)));
    F2=reshape(F1,1,prod(size(F1)));
    x=x2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd)
(3*K+a+3*dd):(3*K+b+3*dd) (4*K+a+4*dd):(4*K+b+4*dd)
(5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)
(7*K+a+7*dd):(7*K+b+7*dd)])
    s=s2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd)
(3*K+a+3*dd):(3*K+b+3*dd) (4*K+a+4*dd):(4*K+b+4*dd)
(5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)
(7*K+a+7*dd):(7*K+b+7*dd)])
    p=p2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd)
(3*K+a+3*dd):(3*K+b+3*dd) (4*K+a+4*dd):(4*K+b+4*dd)
(5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)
(7*K+a+7*dd):(7*K+b+7*dd)])
    v=v2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd)
(3*K+a+3*dd):(3*K+b+3*dd) (4*K+a+4*dd):(4*K+b+4*dd)
```

```
(5*K+a+5*dd):(5*K+b+5*dd)  (6*K+a+6*dd):(6*K+b+6*dd)
(7*K+a+7*dd):(7*K+b+7*dd)])
   T=T2([a:b  (K+a+dd):(K+b+dd)  (2*K+a+2*dd):(2*K+b+2*dd)
(3*K+a+3*dd):(3*K+b+3*dd)  (4*K+a+4*dd):(4*K+b+4*dd)
(5*K+a+5*dd):(5*K+b+5*dd)  (6*K+a+6*dd):(6*K+b+6*dd)
(7*K+a+7*dd):(7*K+b+7*dd)])
   pH=pH2([a:b  (K+a+dd):(K+b+dd)  (2*K+a+2*dd):(2*K+b+2*dd)
(3*K+a+3*dd):(3*K+b+3*dd)  (4*K+a+4*dd):(4*K+b+4*dd)
(5*K+a+5*dd):(5*K+b+5*dd)  (6*K+a+6*dd):(6*K+b+6*dd)
(7*K+a+7*dd):(7*K+b+7*dd)])
   DO2=DO22([a:b  (K+a+dd):(K+b+dd)
(2*K+a+2*dd):(2*K+b+2*dd)  (3*K+a+3*dd):(3*K+b+3*dd)
(4*K+a+4*dd):(4*K+b+4*dd)  (5*K+a+5*dd):(5*K+b+5*dd)
(6*K+a+6*dd):(6*K+b+6*dd)  (7*K+a+7*dd):(7*K+b+7*dd)])
   F=F2([a:b  (K+a+dd):(K+b+dd)  (2*K+a+2*dd):(2*K+b+2*dd)
(3*K+a+3*dd):(3*K+b+3*dd)  (4*K+a+4*dd):(4*K+b+4*dd)
(5*K+a+5*dd):(5*K+b+5*dd)  (6*K+a+6*dd):(6*K+b+6*dd)
(7*K+a+7*dd):(7*K+b+7*dd)])
```

The obtained trajectories are the output of the subprogram. They are used in the main program for calculation of the coordinating variables. The calculations stops when the number of iteration is reached or when the criteria for convergence of the coordinating procedure is satisfied.

6.6. Results from calculations with the Matlab sequential program

The program for sequential calculation is used to calculate the optimal trajectories of the physiochemical(control) and biological(state) variables of the fed-batch fermentation process.

The program is run with the data given in point 6.2 and Table 6.2. The results are given on Fig 6.9 – Fig 6.13.

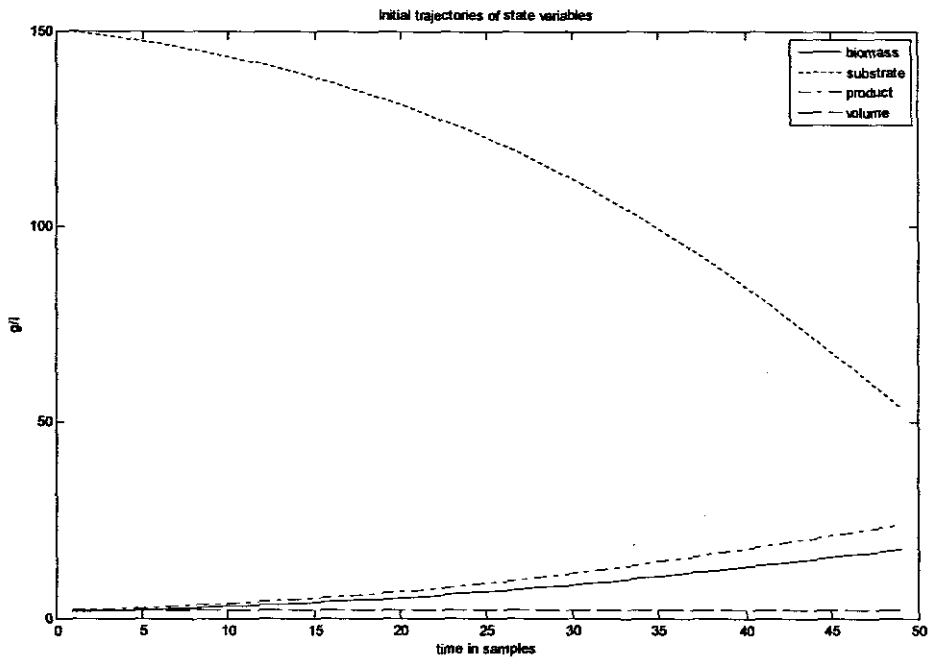


Figure 6.9: Initial trajectories of the state variables

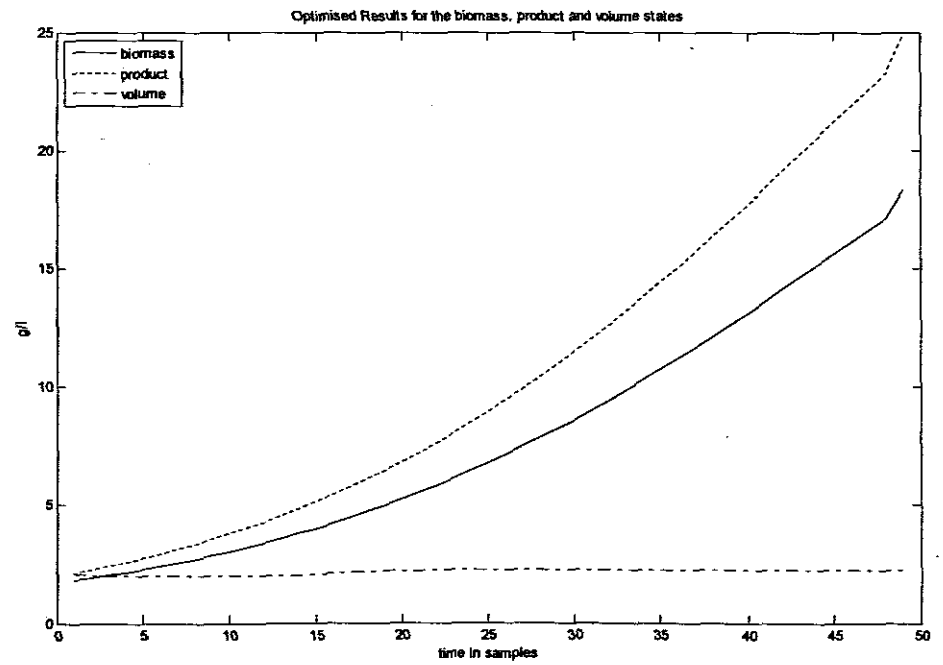


Figure 6.10: Optimal trajectories of biomass product and volume

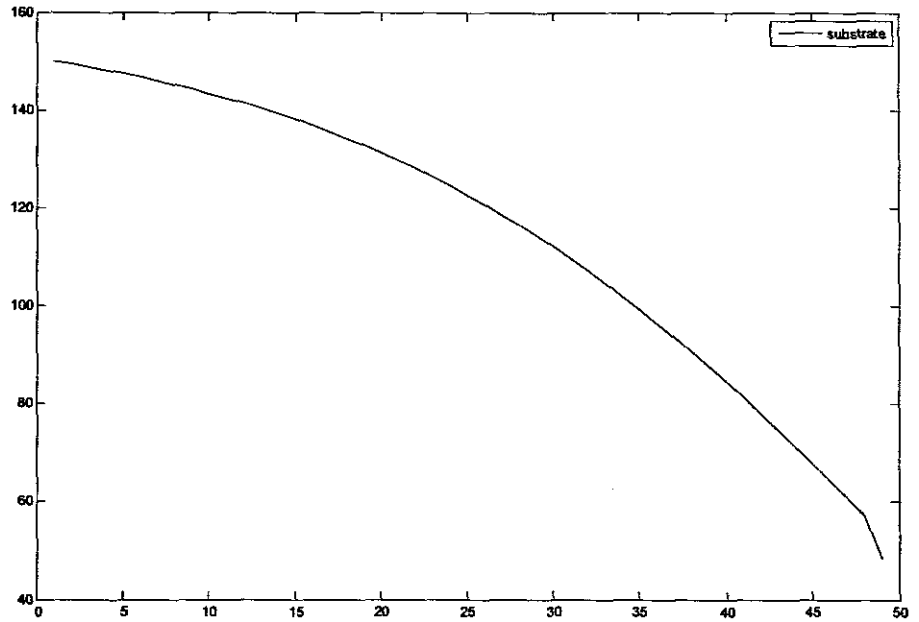


Figure 6.11: Optimal trajectories of substrate

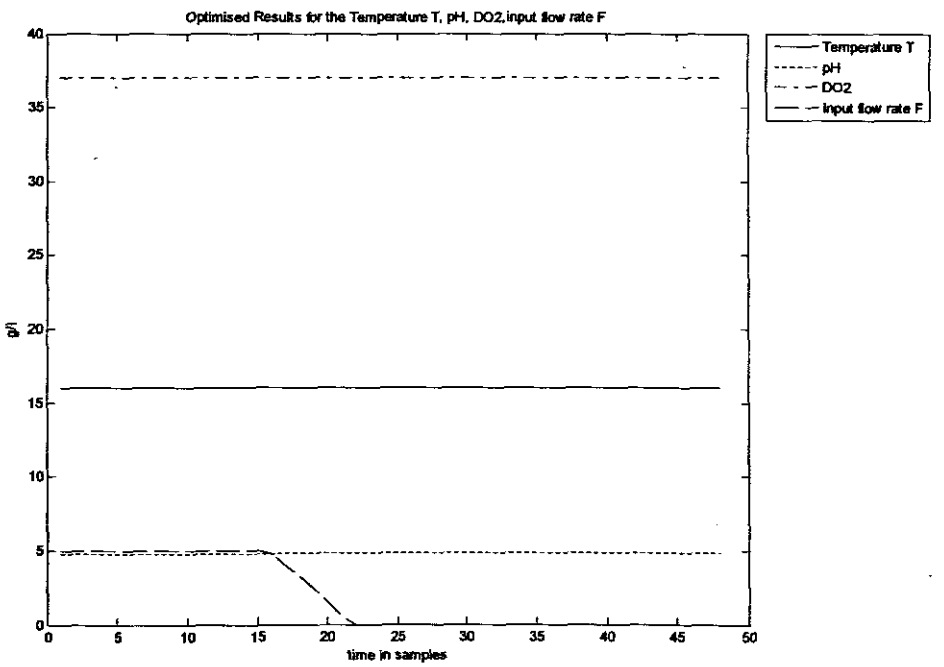


Figure 6.12: Optimal trajectories of the control variables

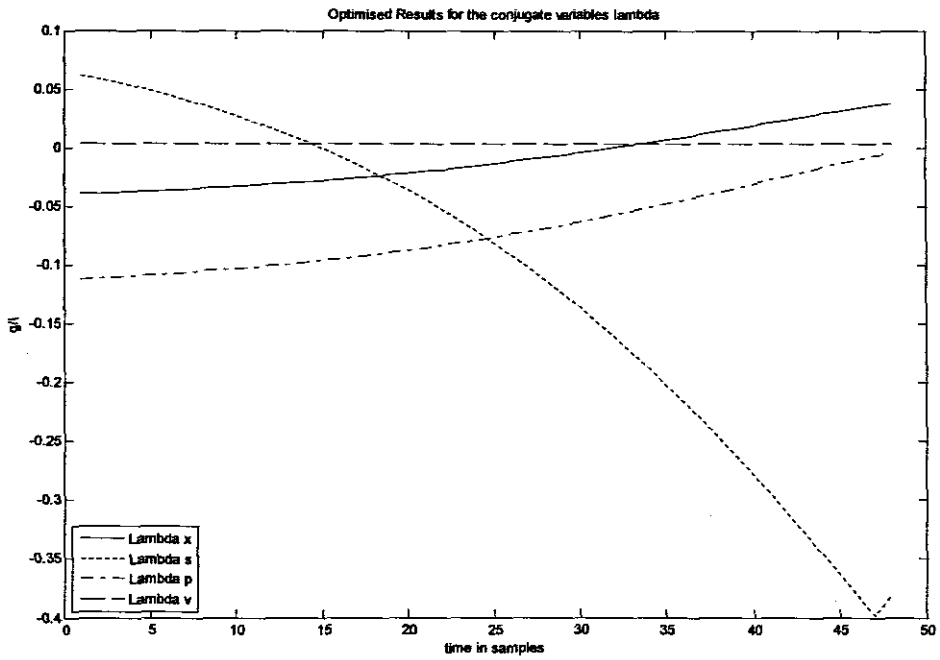
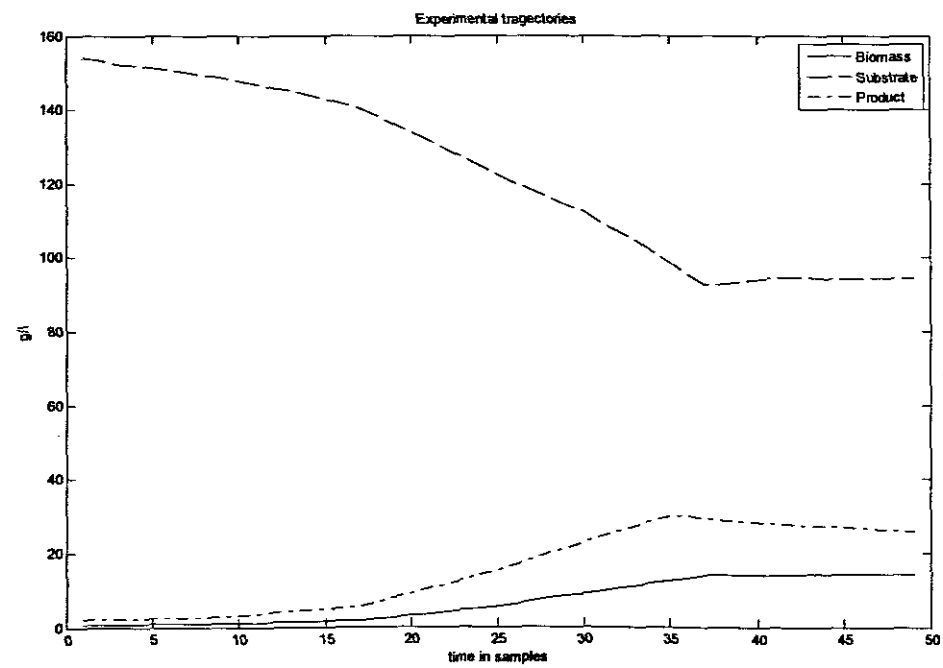
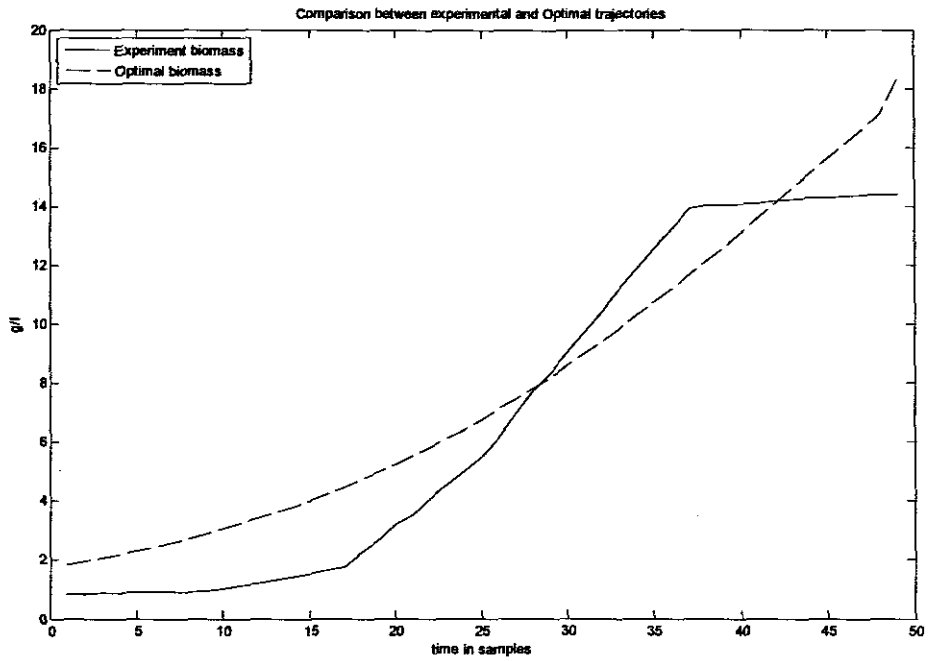


Figure 6.13: Optimal trajectories of the conjugate variables

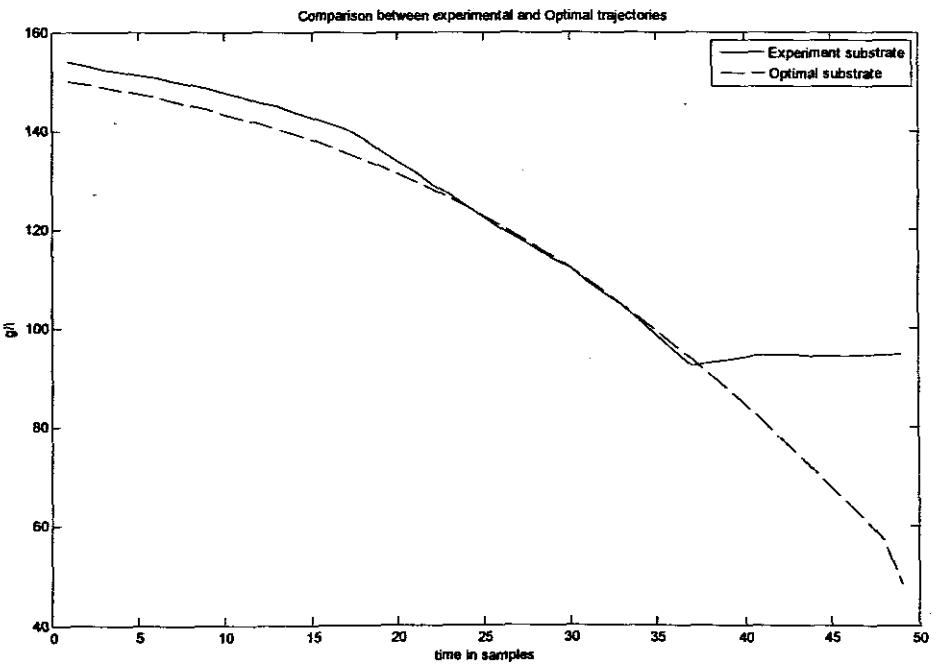
The results are compared with the experimental data given in (Njodzi, 2001) and shown on Fig 6.14(a). The comparison between the experimental trajectories obtained without optimization and the optimal ones obtained by the optimal control calculation is shown on Fig 6.14(b)—Fig 6.14(d).



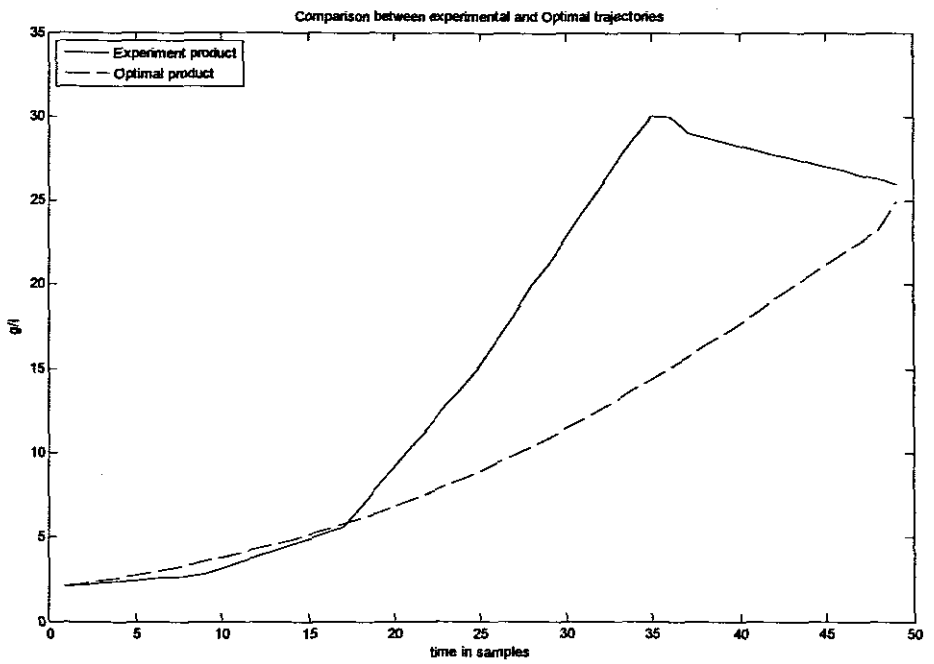
(a)


Comparison between experimental and Optimal trajectories
Experiment biomass
Optimal biomass
g/l
time in samples

(b)


Comparison between experimental and Optimal trajectories
Experiment substrate
Optimal substrate
g/l
time in samples

(c)



(d)

Figure 6.14 Experimental trajectories, Comparison between experimental and optimal trajectories

The aim in production of yeast is to reduce the production of the product of fermentation, ethanol. It can be seen from Fig 6.14(d) that the experimental results with constant trajectories are not very good. The product inhibits the production of biomass as its quantity is very high towards the end of the fermentation. The optimal trajectories of the product shows lower values during the fermentation which allows production of more biomass as can be seen from Fig 6.14(b).

The optimization also leads to better utilisation of substrate Fig 6.14(c) and less requirements for additional supply of it.

6.7. Results from the parallel calculations

The results for the parallel calculation are shown on Fig 6.16 – Fig 6.20.The calculation is done with the same input data and with 8 workers.

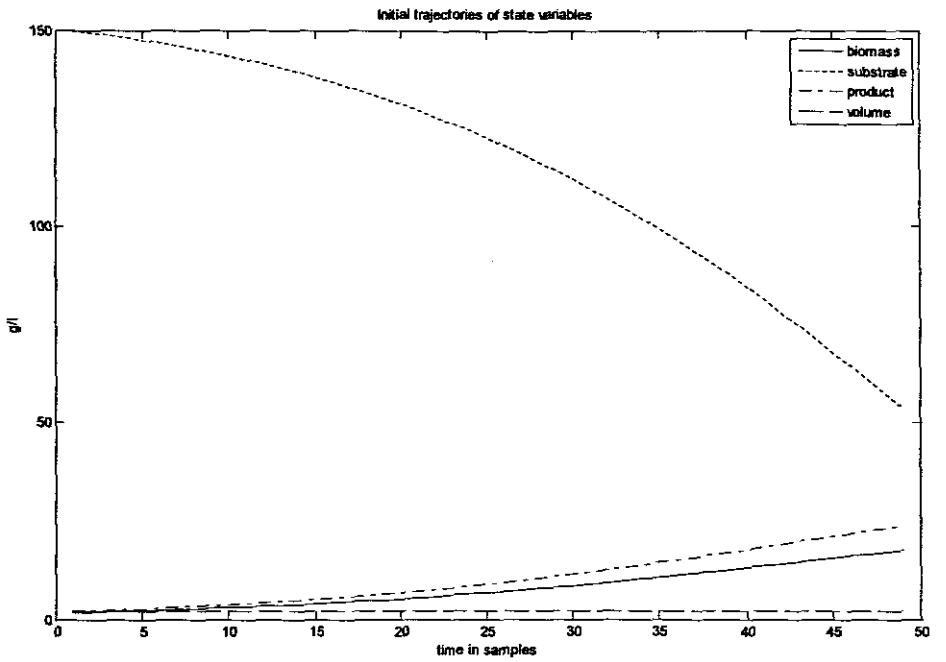


Figure 6.15: Initial trajectories of the state variables

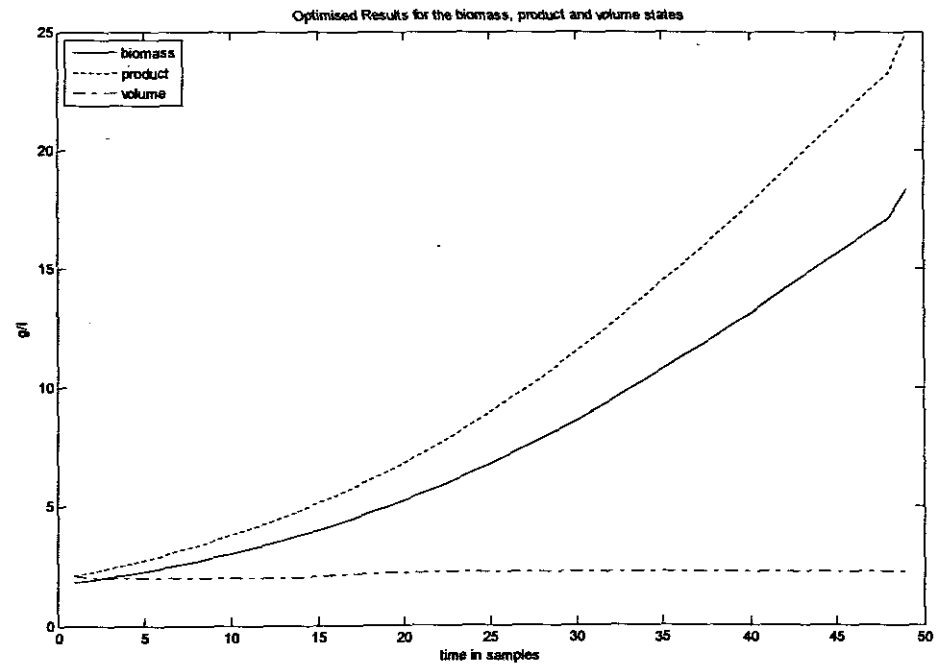


Figure 6.16: Optimal trajectories of biomass product and volume

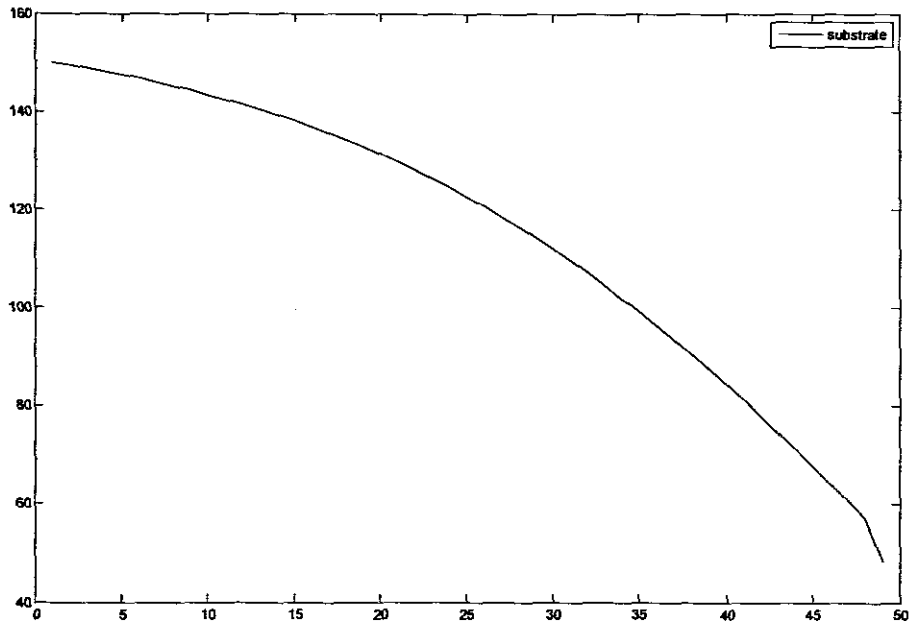


Figure 6.17: Optimal trajectories of substrate

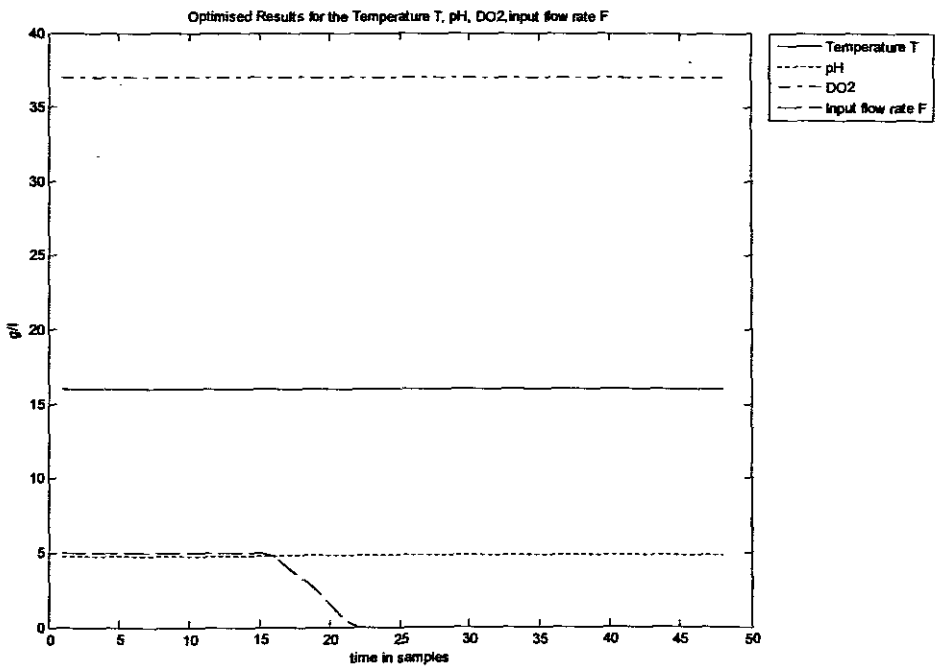


Figure 6.18: Optimal trajectories of the control variables

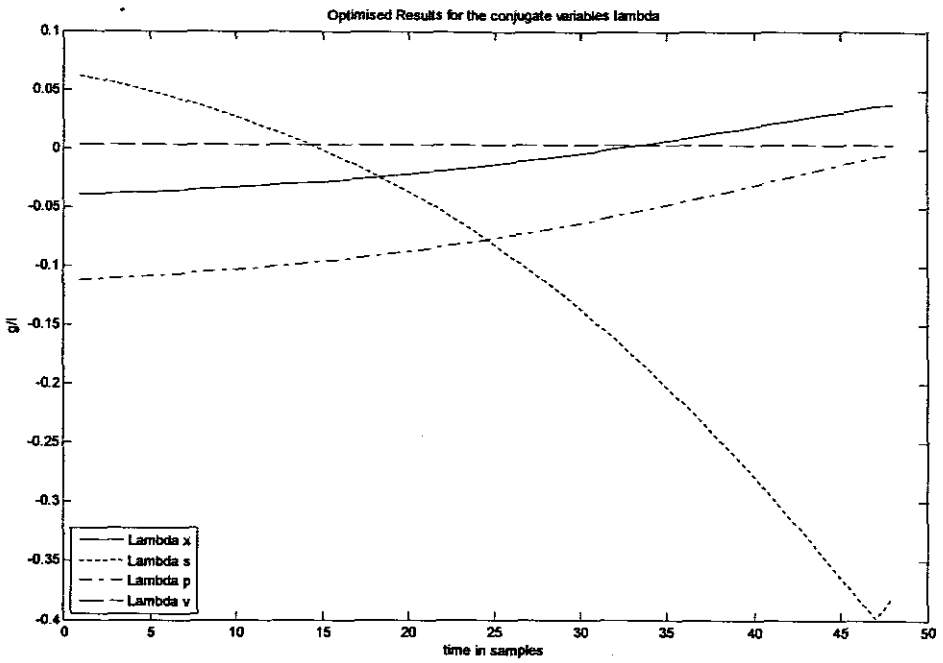


Figure 6.19: Optimal trajectories of the conjugate variables

6.8. Comparison of the results from the sequential and parallel computing

The comparison of the Fig 6.9 – Fig.6.13 and Fig 6.16 – Fig 6.20 shows that the results are the same. This means that the program for sequential computation is transformed properly to a program for parallel computation.

The time used for parallel calculation is less than the time for sequential one. The difference is not big because of the big number of the global variables: Different possibilities for organization of the input data and the data exchanged between the coordinator (the client PC) and the workers to have to be investigated in future investigations.

An investigation of the dependence of the time used for parallel computing from the number of used workers is given in Table 6.2. The time used for every iteration for the corresponding numbers of workers is given in the rows of the table.

* / •	1	2	3	4	5	6	7	8
1	10	9	7	60	8	32	19	25
2	3	6	8	149	13	30	24	13
3	499	306	257	155	29	60	11	12
4	21	65	189	14	28	25	19	12
5	2	101	13	10	20	60	11	13
Total	535	487	471	338	98	207	84	75

* Numbers of used workers
• Iterations

Table 6.2 Time used for optimal control calculation in dependence on the number of workers in the cluster

It can be seen from the Table 6.2 that the time for calculation is reduced with the increase of the number of workers.
This dependence can be seen from the Fig 6.20.

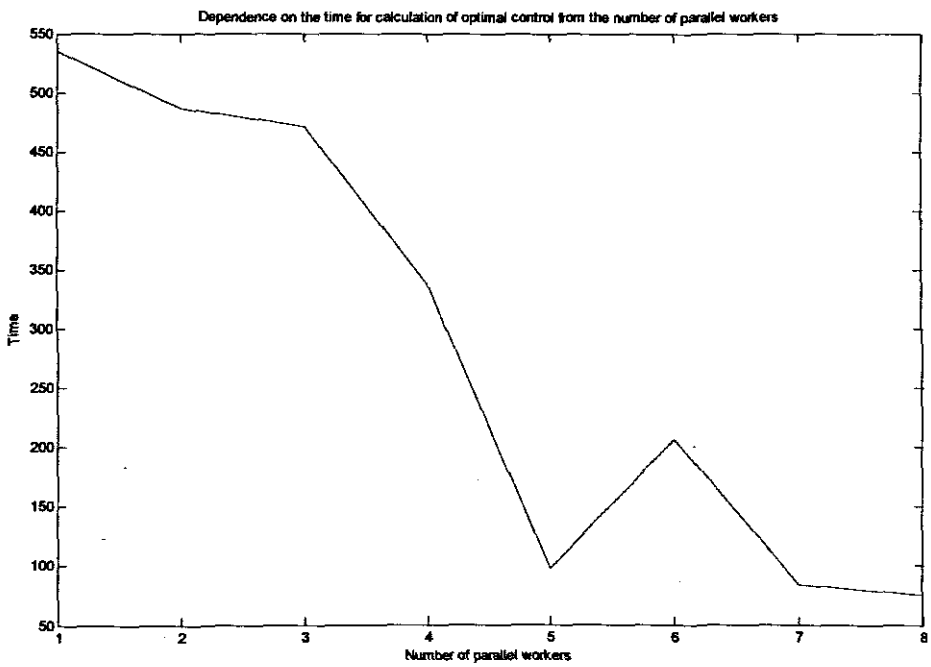


Figure 6.20 Dependence on the time for calculation of optimal control from the number of parallel workers

The successful implementation of the decomposition algorithm and the obtained results show that the decomposition of the problem for optimal control calculation is very convenient to be implemented in a cluster of computers.

6.9. Conclusion

This chapter describes the programmes for optimal control calculation in MATLAB/Simulink and Distributed Computing Toolbox. The input and output variables of the programmes and the used functions and commands are described. The

Results from calculations with the Matlab sequential program, the simulink program and the parallel calculations also are described. The results from the sequential and parallel computing are compared.

Chapter 7
Conclusions

Fed batch fermentation, due to an inherent flexibility of the methods provides a valuable tool in biotechnology, not only to study microorganisms, but also to commercially produce valuable components. However, many requisites and variables should be taken into consideration in its implementation. Moreover, the control of the process is mandatory, which otherwise would render the mode of operation complexity useless. In addition to the complexity of the fermentation, the control mode and parameter to be controlled should also be analysed.

The control of a fed-batch fermentation can implicate many difficulties: low accuracy of on-line measurements of substrate concentrations, limited validity of the feed schedule under a variety of conditions and prediction of variations due to strain modification or change in the quality of the nutrient medium. These aspects point to the need of a fed-batch fermentation control strategy, which is model independent, identifies the optimal state on-line, incorporates a negative feedback control in the nutrients feeding system and contemplates saturation kinetic model, variable yield model, variations in feed substrate concentration and product inhibited fermentation. [Agrawal el all, 1994]

The dissertation contributes to the solution of the problems for modeling and optimal control solutions considered as a part of an adaptive control strategy for control of the yeast production.

7.1 Problems solved in the dissertation

The problems in the dissertation are to develop methods, algorithms and programmes for solution of the problems for modeling and optimal control calculation in a two-layer system for real time optimal control of the Biostat® C pilot plant with the following subsystems:

→ Data acquisition and real time implementation,

→ Modelling and simulation,

→ Model parameter estimation,

→ Process optimisation,

→ PID controller parameter tuning,

7.1.1 Modeling and simulation

- → The fed-batch fermentation process for production of yeast is described as an object of control with input, state space, output and disturbance variables.
- → Model of the fed-batch process based on incorporation of the physiochemical variables into the mass balance equation is developed.
- → The programs in Matlab / Simulink are developed to simulate the model. Simulation can be done with real data from the process, or with selected input data.

The developed model is semi-theoretical in the sense that mass balance equations represent theoretical mathematical model, but its coefficient cannot be calculated mathematically. These coefficients are introduced as functions of physiochemical variables in order to represent the influence of the physiochemical variables over the biological variables. The latter is achieved by representing the kinetic parameters of biological model through quadratic functions of physiological variables. Parameters estimation is used to determine the coefficients of these functions. The utilization of Matlab/Simulink packages made the development much easier when it comes to testing of the prototype.

The results from simulation are used later for model parameter estimation and process optimisation.

7.1.2 Process optimization (optimal control calculation)

- → The problem for maximum production of the biomass is formulated.
- → Method and algorithm to solve the optimal control problem is developed on the basis of augmented functional of Lagrange and decomposition in time domain.
- → Sequential program in Matlab to solve the problem for optimal control is developed.
- → Parallel program in Matlab cluster of computers to solve the problem for optimal control is developed.

The decomposition method used to solve the problem for optimisation (optimal control) allows the overall problem to be decomposed and simplified. The complexity of calculation is reduced by using the parallel calculation of the sub-problems on the first level. The time necessary for the sequential solution.

7.2 Benefits of the two-layer control strategy

Benefits of process optimization and adaptive control come from improved plant and business performance. In general terms, the revenues come from improved yields of valuable products, reduced energy consumption, and higher processing rates either through increased capacity of existing equipment or longer stream factors. Optimization may also influence reduction in a number of other operating costs including maintenance, equipment wear and staff utilization.

The engineering benefits come from improved process troubleshooting and assistance in making quicker and more accurate decisions. This ultimately relates to improved process operations. There is also an interaction with process design of new units. The knowledge that comes from computer optimization can have a bearing on equipment sizing and upon assumed capacity factors.

The positive characteristics of the developed methods, algorithms and programmes are:

- → The developed model incorporates the physiochemical variables in the biological mass balance equations. In this way: the influence of the enzymes over the biological variables is utilised and possibilities for process optimisation is created.
- → The process can be optimised in both physiochemical and biological variables.
- → The physiochemical variables can be used as control inputs to reach the process optimization.
- → The proposed method for solution of the problem of optimal control introduces new coordinating vector for time domain decomposition of the problem. In this way the complexity of the problem is reduced and the solution of the nonlinear two-point boundary value problem is avoided.

- → The introduced decomposition allows naturally application of the parallel computing cluster of computers by which the sub-problems on the optimal control problem are solved in parallel.
- → The parallel solution is for shorter time in comparison with the sequential one.
- → The optimal control of the process can be achieved without using expensive on-line sensors for measurement of the biological variables. This is why the developed system is applicable to the existing hardware and software control and measurement systems in industry.
- → The control system automates the operation of the lab scale fermentation unit. It is safe, stable and operational.

7.3 Future developments of methods and applicability

The future developments can be decomposed in the following way:

- → Testing of the two-layer control system on the working plant
- → Testing of the optimisation program for different experiments
- → Testing of the developed application on the bigger scale fermentor
- → Connection of estimation and optimal control problem solution in order to realise different operator strategies
- → Application of the developed methods and programmes for different fed-batch processes

7.4 Application of developed strategy, methods, algorithm and programs

7.4.1 Implementation of the programs

The developed programs for modelling and for sequential optimal control calculation is implemented in the fermentor Biostat ® C in the laboratory of the department of Chemical Engineering, UCT. The students are doing their practical exams using this fermentor.

7.4.2 Application of results

The developed approach for modeling and for optimal control problem solution can be applied with small modifications to some industrial processes with similar characteristics, as:

- → Waster water purification plants.
- → Beer industry.
- → Food industry.
- → Mining industry for extraction of metal ions.
- → Sugar and pharmaceutical industry.
- → Control purposes in education.
- → As an application plant for different control strategies.

7.5 Publications in connection with the dissertation

- Chen. H, R. Tzoneva, Optimal control of a fed-batch fermentation process for production of yeast, sent to the journal SAIEE Transactions, 2006.

References:

AGRAWAL P., KOSHY G., RAMSEIER M., 1989: An algorithm for operating a fed-batch fermentor at optimum specific growth rate. Biotechnology and Bioengineering, vol. 33, pp. 115-125.

AGRAWAL P., KOSHY G., RAMSEIER M., 1989: An algorithm for operating a fed-batch fermentor at optimum specific growth rate. Biotechnology and Bioengineering, vol. 33, pp.115-125.

AHMED S. FAWZY, OLIVER R. HINTON, 1984: Optimal and hierarchical control of fermentation processes using a microcomputer: A survey, IEEE, IEEE TRANS. SYST. MAN CYBER. Vol. SMC-14, no. 1, pp. 162-165.

ANDERSON M., BRABRAND H., JORGENSEN S., 1991: Model based control of continuous yeast fermentation, Proc. Of the ACC, 01, Boston, USA, pp. 1329-1334.

ANDRES-TOO B. DE, GIRON-SIERRA J.M., LOPEZ-OROZCO J.A., FERNANDEZ-CONDE C, PEINADO J.M., GARCIA-OCHOA F., 1998: A kinetic model for beer production under industrial operational conditions, Mathematics and Computer Simulation, vol.48, pp. 65-74.

ANTHONY SULISTIO, CHEE SHIN YEO, RAJKUMAR BUYYA, 2004: *A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools*, Software—Practice And Experience. Vol.34, pp. 653–673.

B&B ELECTRONICS, 1997: RS-422 and RS-485 Application Note. www.bb-elec.com.

B. BRAUN BIOTECH INTERNATIONAL, 1996: Installation handbook 1.1 MFCS/win.

B. T. SMITH, J. M. BOYLE, J. J. DONGARRA, B. S. GARBOW, Y. ILEBE, V. C. KELMA, C. B. MOLER, 1976: *Matrix Eigensystem Routines—EISPACK Guide*, 2nd ed. New York: Springer-Verlag.

BAGRODIA R, MEYER R, TAKAI M, CHEN Y, ZENG X, MARTIN J, PARK B, SONG H. PARSEC, 1998: A parallel simulation environment for complex systems. IEEE Computer; vol. 31(10), pp. 77-85.

BAJPAI R., LUECKE R., 1990: Controller design for biochemical reactor, *Chemical Engineering Problems on Biotechnology*, SHULER H.L. Eds, American Institute of Chemical Engineering, NY, vol. 1, pp.301-332.

BARKER M, BUYYA R. CLUSTER COMPUTING AT A GLANCE. *High Performance Cluster Computing*, vol. 1, Buyya R (ed.). Prentice- Hall: Upper Saddle River, NJ, 1999; 3–47.

BASTIN G., 1991: Non-linear and Adaptive Control in Biotechnology: A tutorial, European Control Conference (ECC-91), Vol.3, pp. 2001-2012.

BASTIN G., DOCHAIN D., 1990: *On-line estimation and adaptive control of bioreactors*. Elsevier science Publishing Co. Oxford, UK.

BELLGARDT K., JUAN J., 1991: Optimisation of yeast production – A case study Biotechnology, vol. 4, pp. 383-406.

BIROL G., DORUKER P., KIRDAR B., ONSON, Z., VLGEN K., 1998: Mathematical description of ethanol fermentation by immobilized Saccharomyces cerevisiae, Process Biochemistry, vol. 33, No. 7, pp. 763-771.

Bridge VIEW™ and LabVIEW™ NATIONAL INSTRUMENTS, 1998: PID Control Toolkit for G Reference Manual, National Intruments.

C. MOLER. (1995) Why there isn't a parallel MATLAB. [Online]. Available: http://www.mathworks.com/company/newsletter/pdf/spr95cleve.pdf .

CHEN L., BASTIN G., BREUSEGEM V., 1992: A case study of adaptive non-linear regulation of fed-batch biological reactors. Technical report, 92.37. Catholikc universitat, Louvam, Belgium.

CHERUY A., DURAND A., 1979: Optimisation of erythromycin biosynthesis by controlling pH and Temperature: Theoretical Aspects and Practical Application, Biotechnol Bioeng Symp., vol. (9), pp. 303-20.

CLAES J., VAN IMPE J., 1998 (a): On-line monitoring and optimal adaptive control of the fed-batch baker's yeast fermentation. *Preprints of the 7th International Conference on Computer Application in Biotechnology* – CAB VII IT. Yoshida, S. SHIOYA, Eds, pp.405-410, Elsevier science Publishing Co., Osana, Japan.

CLAES J., VAN IMPE J., 1998b: Sensitivity analysis of a linearising controller for the fed-batch baker's yeast fermentation. Technical report Biotech. Department of food and microbial technology, Katholiene universiteit Leuven, Belgium.

CLAES J., VAN IMPE J., 1999: Sensitivity analysis of a linearising controller for the fed-batch baker's yeast fermentation. Proc. Of the 14th world congress of IFAC, vol. o, pp.439-444.

DESHPANDE, P.B & CHEN, L.C. 1993: Real Time Simulation and Advanced Process Control, Present Status and Future Trends. Proceedings of the 19th Annual Advanced Control Conference. Indiana: W.Lafayette, pp.1156-1159.

Distributed Computing Toolbox For Use with MATLAB®, 2005 http://www.mathworks.com/access/helpdesk/help/techdoc/learn_matlab/learn_matlab.html .

DOCHAIN D., BASTIN G., 1985: Stable adaptive algorithms for estimation and control of fermentation processes, Proc. 1st IFAC Symposium on Modelling and Control of Biotechnological Processes, Noordwijckerhout, The Nether-lands, pp. 1-6.

DOCHAIN D., BASTIN G., 1990: Adaptive Control Of Fed-batch Bioreactor, Chemical Engineering Comm., vol. 87, pp.67-85.

DORAN PAULINE M, 1995: Bioprocess Engineering Principles Chapters 4, 6 & 11.
POMERLEAN Y., VIEL G., 1992: Industrial application of adaptive non-linear control for baker's yeast production. Proc of IFAC symp. on modeling and control of biotech, processes, key stones, USA, pp.315-318.

DORF, R.C. 1998: *Modern Control Systems*, 8th Edition, Addison Wesley.

DUBE, N.M., 2002: *Modelling and optimal control of counter current ion exchange process*. M-Tech thesis, Peninsula Technikon.

ECHEGARAY O.F., CARVALHO J.M., FERNANDES A.N.R., SATO S., AQUARONE E., VITONLO M., 2000: Fed-batch culture of Sacchoromyces cerevisiae in sugar-cane blackstrap molasses: invertase activity of intact cells in ethanol fermentation, Biomass and Bioenergy, Vol. 19, 39-50.

FINDEISEN, W. 1980: *Control and Coordination in Hierarchical Systems*. Chichester, West Sussex: Wiley.

FISHER, W, DOHERTY, M. & DOUGLAS J. 1984: An Evolution and Hierarchical Procedure for Optimization of Preliminary Process Design. Annual Meeting of AICHE, Paper 104c. San Francisco, pp. 88-92.

FLAUS J.M., PONS M.N., CHERUY A., and ENGASSER J.M., 1989: *Adaptive algorithm for estimation and control of fed-batch ioprocess*, Prectice Hall, NY.

FOSTER I, KESSELMAN C (eds.)., 1999, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann: San Francisco, CA.

FREDE LEI, MORTEN ROBOLL, and STEN BAY JORAGENSEN, 2001: A bio-chemically structured model for Saccharomyces cerevisiae, J Biotechnol; vol. 88(3), pp. 205-21.

GARCIA A., ACEBES L.FL., and C. de PRADA, 2002: Modelling and simulation of batch processes: A case study. Proc of the 15th Triennial world congress of the international federation of automatic control, Barcelona, IFAC, CD-ROM.

GEORGIEV TZ., KRISTRVA, V. IVANOVA, B. RATKOV, RATKOV AI. (2003), Comparative Sudies of Fed-Batch Fermentation Processes for Production of L-lysine and L-valine Based on Mathematical, In: Proc. 11th European Congress on Biotechnology, Basel, 304, pp. 112-113.

GEORGIEV TZ., T. VARBEV (2003), Failurs of Power Supply System of Open-air Mine "Hristo Botev" of Mines "Bobov Dol" *Ltd.,* In: *50 yers University of Mining and Geology*: St. Ivan Rilski", Annual vol 46, Part III, Mechanization, Electrification and Automation of Mines, Sofia, 2003, 125-127.

GOU S., JUAN J., BELLGARDT., 1995: Profit optimization for baker's yeast continuous fermentation, 6th International Conference on Computer Application in Biotechnology, Garmisch Partenkirchen, Germany, pp. 168-176.

GYEONGBEOM YI, YOUNG BO HWANG, HO NAM CHANG and KWANG SOON LEE, 1989: Computer Control of Cell Mass Concentration in Continuous Culture, Proc. of the IFAC world congress, pp. 243-249.

HASKINS, D. 1983: Restraints on Entire Plant Optimization. Chemical Engineering Process, vol. 79, No.6, pp. 213-218.

HIGGIN, J. 1984. Process Control No Longer Separate from Simulation Design. Chemical Engineering News, vol. 62, No. 14, pp. 286-291.

HONGWEI ZhANG, BARRY LENNOX (2003), Integrated condition monitoring and control of fed-batch fermentation processes, Transactions of the Institute of Measurement and Control, vol. 25 (5), pp. 403-417.

HORIUCHVI J., KAMASAWA, M., MIYANAWA H., KISHIMORO M., MOMOSE H., 1993: J. fermentation bioengineering, vol. 76, pp.207-211.

RS232/RS485 Converter, Users Manual, http://www.aten.com.tw/ .

HWANG Y., DOCHAN D., BASTIN G., 1995: Performance of inhibition functions with total inhibition concentration. Proceedings of the 6th International Conference on Computer Application in Biotechnology, Garmisch Partenkirchen, Germany, pp.259-262.

Ian Foster, Designing and Building Parallel Programs, http://www-unix.mcs.anl.gov/dbpp/text/book.html .

IGNOVA M., PAUL G.C., KENT C.A., THOMAS C.R., MANTAGUE G.A., GLASSEY J., and WARD A.C., 2002: Hybrid modeling for on-line penicillin fermentation optimization. Proc of the 15th Triennial world congress of the international federation of automatic control, Barcelona, IFAC, 21-26 July 2002, CD-ROM.

ILSE SMETS, KRISTEL BERNAERTS, JUN SUN, KATHLEEN MARCHAL JOS BANDERLEYDEN and JAN VAN IMPE, 2002: Sensitivity function based model reduction. Proc of the 15th Triennial world congress of the international federation of automatic control, Barcelona, IFAC, 21-26 July 2002, CD-ROM.

ISIDORI, A. 1995: *Nonlinear Control Systems*. London: Springer.

J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, 1979, *LINPACK User's Guide*. Philadelphia, PA: SIAM.

JENS E. HAAG, ALAIN VANDE WOUWER, and MARCEL REMY, 2002: S systematic model reduction procedure for complex bioprocesses described by metabolic pathway networks. Proc of the 15th Triennial world congress of the international federation of automatic control, Barcelona, IFAC, 21-26 July 2002, CD-ROM.

JIN S. K. Ye, SIMIZU K., CHEN J., 1994: Tech. Biotech, vol. 61, pp.273-281.

JOHANSSON, R, 1993: *System Modelling and Identification*. New York: Prentice Hall.

JOHNSON A., 1993: The theory of LQ optimisation of fermentation. Proc. Of the X11th congress IFAC congress, vol. 9, pp. 115-119.

JOHNSON, C.D. 1993: *Process Control Instrumentation Technology*, 4th Edition. New Jersey: Prentice Hall Inc.

JONES, L.D. 1991; *Electronic Instruments and Measurements*, 2nd Edition. New Jersey: Prentice Hall Inc.

JORGENSEN S., JENSEN N., 1989: Dynamics and control of chemical reactorselectivity surveyed. Proc. Of DYCORD'89, Maastiicht, Holland, pp. 156-160.

JUNKINS, J.L. 1991: *An Introduction of Optimal Estimation of Dynamic Systems*. Alphen Aan Den Rijn: SIJHOFF and NOORDHOFF INTERNATIONAL.

KARIM M., RIVERA S., 1992: Adv. Biochemical. Engineering / Biotech. Vol. 76, pp.1-10.

KEULERS M., ARIAANS L., GIUSEPPIN M., SOETERBOEK R., 1993: Specific growth rate control in fed- batch baker's yeast fermentation. Proc of the X11 IFAC world congress, vol. 7, pp. 441-444.

KLEMAN G., CHALMERS J., LULI G., STROHL W., 1991: A predictive and feedback control algorithm maintains a constant glucose concentration in fed-batch fermentations. Applied and Environment Microbiology, vol. 57, pp.910-917.

KOLEV N. P., S. T. YORDANOVA, P. M. TZVETKOV (2002), Computerised Investigation of Robust Measurement Systems, IEEE Trans. Instrumentation and Measurement, vol. 51, No 2, April 2001, ISSN 0018-9456, 207-210.

KONSTANTINOV K., YOSHIDA T., 1989: Biotech Bioengineering, vol. 33, pp.1145-1150.

KOZIETULSKI, M. 1981: Two-Layer Implementation of Repetitive Dynamic Optimization (Static-Optimally Aided Control). Archivum Automatiki I Telemeckaniki, vol. 26, No. 4, pp.511-520.

LEFKOWITZ, 1996: Multilevel Approach to Control System Design. Transcript of ACME, Journal of BE, vol.6., pp. 98-101.

LIPTAK, B.G. 1999: *Instrument Engineer's Handbook. Process Control*, 3rd Edition. CRC Press LLC.

Liu X, Liu J, Eker J, Lee EA, 2003 *Heterogeneous modeling and design of control systems. Software-Enabled Control: Information Technology for Dynamical Systems*, Samad T, Balas G (eds.). Wiley-IEEE Press: New York.

LUYBEN, W.L. 1990: *Process Modelling, Simulation and Control for Chemical Engineers*, 2nd Edition. McGraw Hill Publishing Co.

M O TOKHI, FIAZ H HUSSAIN, M A HOSSAIN, M H SHAHEED, *Parallel Computing for Real-Time Signal Processing and Control*, Springer-Verlag, Published March 2003.

M. Frigo and S. Johnson, "FFTW: an adaptive software architecture for the FFT," in Proc. Int. Conf. Acoustics, Speech and Signal Processing, vol. 3, 1998, p. 1381.

MATLAB Distributed Computing Engine® For Use with MATLAB, 2005 http://www.mathworks.com/access/helpdesk/help/pdf_doc/mdce/mdce.pdf . https://tagteamdbserver.mathworks.com/ttserverroot/Download/29492_91263v01_DCT_datasheet.pdf .

MC NEIL B., HARVEY L., 1990: *Fermentation, a practical approach*, IRL Press, Tokyo.

Measurements within the Framework of State Estimation in the Presence of Persistent Unknown Disturbances. AIChE,(American Institute for Chemical Engineers) vol.26, no.2, pp.247-260.

MESAROVIC, M.D. 1970: *Theory of Hierarchical, Multilevel Systems.* New York: Academic Press.

MESZAROS A., BALES V., 1992: A contribution to optimal control of fed-batch biochemical processes. Bioprocess Engineering, vol. 7, pp.363-367

MICHAEL L. SHULER, FIKRET KARGI, 1992: Bioprocess Engineering Basic Concepts

MITKO PETROV, TATJANA ILKOVA, STOYAN TZONKOV, ULDIS VIESTURS; Modelling, Optimization and Optimal Control of Small Scale Stirred Tank Bioreactors; Bioautomation, 2004, vol. 1, pp. 67-82.

MORARI, M. & STEPHANOPOULOS, G. 1980: Studies in the Synthesis of Control Structures for Chemical Process. Part II: Structural Aspect and the Synthesis of

Alternative Feasible Control Schemes. AIChE, (American Institute for Chemical Engineers) vol.26, no.2, pp.232-247.

MORARI, M. & STEPHANOPOULOS, G. 1980: Studies in the Synthesis of Control Structures for Chemical Process. Part III: Optimal Selection of Secondary.

MORARI, M., ARKUN, J. & STEPHANOPOULOS, G. 1980: Studies in the Synthesis of Control Structures for Chemical Process. Part I: Formulation of the Problem. Process Decomposition and the Classification of the Control Tasks. Analysis of the Optimizing Control Structures . AIChE, (American Institute for Chemical Engineers) vol.26, no.2, pp. 220-232.

MOSER A., 1985: *Kinetics of batch fermentation in biotechnology*, Ed. H. REHM and REED G. VCH verlagsgesellschaft mgh, Weinhem, pp.273-283.

NAKAMURA T., KURATANI T., MORITA Y., 1985: Proc of the IFAC Modelling and control of biotechnology. Processes, Noordwijkerhout, pp.231-235.

NEWAY J., 1989: Fermentation process development of industrial organism, Narcd Dekker, NY.

NIKOLOVA N., E. NIKOLOV (2004), Fractional Robust Control System with Internal Model, In: Proc. of International Conference AUTOMATICS AND INFORMATICS'04 Sofia, October 6-8, 2004, ISSN 954-9641-39-2, Vol. 1, © 2004 Union of Automation and Informatics, pp. 171-174.

NJODZI Z., 2001: *Studies on the fed-batch propagation of brewer's yeast in high gravity wort*, Msci assertation, UCT. Cape Town.

NORTON, J.P. 1986: *An Introduction to Identification.* London: Academic Press, Harcourt Brace Javanovic Publishers.

OGATA K., 1990: *MODERN CONTROL ENGINEERING*, second edition, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

PARK Y., SHI Z, SHIBA S., CHANRAL C., RIJIMA S., KOBAYASHI T., 1993: Application, Microbial. Biotechnology. vol. 38, pp.649-655.

PARTNAIK P., 1995: A heuristic approach to fed-batch optimization of streptokinase fermentation. Bioprocess Engineering, vol. 13, pp.109-112.

PASCAL SIEGWART, JOHANNE COTE, KEITH MALE, JOHN H.T. LUONG, MICHEL PERRIER, and AMINE KAMEN: Adaptive Control at Low Glucose Concentration of HEK-293 Cell Serum-Free Cultures, Biotechnol Prog. 1999 Jul-Aug; vol. 15(4), pp. 608-16.

POMERLEAU Y., PERRIER M., BOURQUE D., 1995: Dynamics and control of the fed-batch production of poly – β – Hydroxybutyrate by Methyl bacterium extorquens

POPCHEV, T. & TSONEVA, R. 1990: Synthesis of Tow-Layer Control of Interconnected Systems with Time Delays in the Presence of Slowly Varying Disturbance. Proceedings for the, Tallinn: XI World Congress IFAC, vol. 2, pp. 25-31.

REULERS M., ARIAANS L., GIUSEPPIN M., SOETERBOEN R., 1993: Specific growth rate control in fed-batch baker's yeast fermentation. Proc. Of the X11th congress if IFAC, vol. 4, pp.441-444.

R. CLINT WHALEY, A. PETITET, and J. J. DONGARRA, 2001,"Automated empirical optimization of software and the ATLAS project," Parallel Comput., vol. 27, no. 1–2, pp. 3–35.

ROBERTS, P.D. & LIN, J. 1991: Potential for Hierarchical Optimizing Control in the Process Industry. Proceedings of the 1st European Control Conference, Grenoble, France, vol. 1,pp.213-217.

ROEL A., 1983: *Enerpetics and kinetics in biotechnology*, Elsevier, Biomedical Press, Amsterdam.

RON CHOY, ALAN EDELMAN, 2005, Parallel MATLAB: Doing it Right, Computer Science AI Laboratory.

ROSIMEIRE APARECIDA JERONIMO, CLAUDIO GARCIA, and OSCAR A.Z. SOTOMAY OR, 2002: Fuzzy identification of bioprocesses applying TSK-TYPE models with consequent parameter estimation through orthogonal estimator. Proc of the 15th Triennial world congress of the international federation of automatic control, Barcelona, IFAC, CD-ROM.

ROUBOS J. A., DE GOOOIJIER C. D., VAN STRATEN G., VAN BOXTEL A.J.N, 1997: Comparison of optimization methods for fed-batch cultures of hybridoma cells, Bioprocess and Biosystems Engineering, Springer-Verlag GmbH, pp. 99-102.

SHI Z., SHIMIZU K., 1992: J.Ferm. Bioeng. Vol. 74, pp. 39-75.

SHIMIZU K. K. Ye, 1995: Development of intelligent control systems for bioreactors. Preprints of the 6th International Conference on Computer Applications in Biotechnology, Garmish Patennirchen, Germany, pp. 89-94.

SHIMIZU KAZUYU And KAIMING YE,1995: Development of intelligent control systems for bioreactors, Computer Applications in Biotechnology, May, pp.14-17.

SHIMIZU N., MIURA K., SHIOYA S., SUGA K., 1994: Proc of PSE'94, Kyongju, Korea.

SHIOYA S., SHIMIZU H., OGATA M. And TAKAMATSU T., 1985: Simulation and experimental studies of the profile control of the specific growth rate in a fed- batch culture, IFAC Conference on Modelling and Control of Biotechnological Processes, pp. 493-501.

SKOGESTAK, S.: HALVORSEN, I: LARSSON, T & GOVATSMSRK, M.1999: Plant-Wide Control: The Search for the self-optimizing Control Structure. Proceedings of the 14th IFAC World Congress, China, pp. 325-330.

STANDBURY P., WHITAKER A., HALL S., 1993: *Principles of fermentation technology*, Pergamon Press, NY.

TAMURA, H. 1975. Decentralized Optimization for Distributed Lag Models of Discrete Systems, Automatica, vol. 11, pp. 593-602.

TARTAKOUSKY B., VLITZUR S., SHEINTUCH M., 1995: Optimal control of fed-batch fermentation with autoinoluction of metabolite production. Biotechnology Prof. vol. 11, pp.80-87.

TATIANA ILKOVA, STOYAN TZONKOV; Optimal Control of a Fed-batch Fermentation Process by Neuro-Dynamic Programming, Bioautomation, 2004, vol. l, pp. 57-66.

TEO, K.L.; GOH, C.J. &WONG, K.H. 1991: *A Unified Computational Approach to Optimal Control Problems*. London: Longman.

TURNER C., GREGORY M., THNNHILL N., 1994: Closed loop control of fed-batch cultures of recombinant Escherichia coli using on-line HPLC. Biotechnology and Bioengineering, vol. 44, pp.819-829.

UNBAHANEN, H & RAO, G. 1997: Continuous Time Approach to System Identification: A Tutorial in Preprints of 11th IFAC Symposium on System Identification, vol. 3, pp. 1023-1049.

VAN IMPE J. F. and BASTIN G., 1995 (b): Optimal adaptive control of fed-batch fermentation processes. Cont. Eng. Pract. vol. 3, No.7, pp.939-954.

WILLIAMS D., YOUSEFPOUR P., WELLINGTON E., 1986: On-line adaptive control of a fed-batch fermentation of Saccharomyces cerevisiae. Biotechnology and Bioengineering, vol. 18, pp.631-645.

YU, W. & LIU, D. 1999: Time Variant Parameter estimation of a Nonlinear System using a Quasi-Newton Method. Proceeding of 14th world Congress of IFAC, China, vol. H, pp.55-60.

ZENG X, BAGRODIA R, GERLA M. GLOMOSIM, 1998: A library for parallel simulation of large scale wireless networks. Proceedings 12th Workshop on Parallel and Distributed Simulations (PADS98), Banff, Alberta, Canada, May 1998. IEEE Computer Society Press: Los Alamitos, CA; 154–161.

Appendix A

1. Program optconfer.m for sequential calculation of the optimal trajectories.

(9 pages)

2. Program model_func1 for calculation of the derivatives of the Biomass variables.

(1 page)

```
clear all
%close all

tic;
global par T pH DO2 F kd m pl n si
%Set Initial values
M1 = 5; M2 = 5;%Max number of iterations of the first and second  levels
K = 48;%Number of steps in optimization horizon

ax = 0.001;as = 0.001; ap = 0.001;av = 0.001;%Steps of the gradient procedures for
calculation of the State variables
aT = 0.05; aF = 0.05; apH = 0.05; aDO2 = 0.05;%Steps of the gradient procedures for
calculation of the Control variables
alx = 0.001; als = 0.001; alp = 0.001; alv = 0.001;%Step of the gradient procedures
for calculation of the Conjugate variables
adx = 0.001; ads = 0.001; adp = 0.001; adv = 0.001;%Steps of the gradient procedures
for calculation of the Interconnections in time

eps1=0.001;
eps2=0.001;%tolerances for stpo of the calculation of the first and second levels
%epsx = 0.01; epss = 0.01; epsp = 0.01; epsv = 0.01;%Conjugate error variables
%epsF = 0.01; epsT = 0.01; epspH = 0.01; epsDO2 = 0.01;%Control error variables

%Initial values of the penalty coefficients
mx=0.0010;
ms=0.0010;
mp=0.0010;
mv=0.0010;

%Initial trajectory values of the conjugate and control variables

for k = 1:K
 %Initial trajectories for the conjugate variables
        lx(k)=0.001;
        ls(k)=0.005;
        lp(k)=0.002;
        lv(k)=0.003;
 end
%Initial trajectories of the control variables
T=16;
pH=4.8;
DO2=37;
F=0.;


%F = [zeros(1,40) 0.5+0.05*ones(1,10)]%input flow rate F(k)

%t=1:K
%t1=t';
%pH1=pH';
%T1=T';
%O1=DO2';
%FF=F';
```

```
% Model co-efficients

%a1 = 0.08; a2 = 0.03;a3 = 0.01; a4 = 0.02; a5 = 0.002; a6 = 0.001; a7 = 0.002;
a1 = 0.0004; a2 = 0.0001;a3 = 0.0002; a4 = 0.0001; a5 = 0.0001; a6 = 0.00002; a7 =
0.00002;
%b1 = 0.2; b2 = 0.03;b3 = 0.01; b4 = 0.01; b5 = 0.001; b6 = 0.002; b7 = 0.001;
b1 = 4; b2 = 0.5 ;b3 = 0.13; b4 = 0.13; b5 = 0.013; b6 = 0.014; b7 = 0.013;
%c1 = 2; c2 = 0.3; c3 = 0.021; c4 = 0.051; c5 = 0.00211; c6 = 0.0211; c7 = 0.002;
c1 = 0.0002; c2 = 0.0001; c3 = 0.000021; c4 = 0.00021; c5 = 0.000211; c6 =
0.0000211; c7 = 0.0002;
%d1 = 0.0040; d2 = 0.006; d3 = 0.001; d4 = 0.005; d5 = 0.0005; d6 = 0.001; d7 =
0.0002;
d1 = 0.0360; d2 = 0.008; d3 = 0.003; d4 = 0.005; d5 = 0.005; d6 = 0.001; d7 =
0.0002;
par = [a1 a2 a3 a4 a5 a6 a7 b1 b2 b3 b4 b5 b6 b7 c1 c2 c3 c4 c5 c6 c7 d1 d2 d3 d4 d5
d6 d7];


kd = 0.00008;
pl = 50;
m  = 0.11;
n  = 0.52;
si= 266.6;
dt = 2.5;%Delta t

%Initial values of state variables
x0=1.83;
s0=150.0;
p0=2.1;
v0=2.1;
y0=[x0 s0 p0 v0]';

%Constraints of state and control variables
Tmin = 10;   %%%%%%%%%%%%%%%%%%%%%%%%%
pHmin = 4;  %                         %
DO2min = 10; %                         %
Fmin = 0;    % For control variables%
Tmax = 20; %                         %
pHmax = 7; %                         %
DO2max = 60;%                         %
Fmax = 5.0;  %%%%%%%%%%%%%%%%%%%%%%%%%
% State variables%
vmin = 2;
vmax = 4;
pmin = 2.0;
pmax = 70;
smin = 20;
smax = 300;
xmin = 0.83;
xmax =55;



%Calculation of the initial trajectories of state variables
```

```
t=1:K+1
[t,y]=ode15s('model_func1',t,y0)
x=(y(:,1))'
s=(y(:,2))';
p=(y(:,3))';
v=(y(:,4))';


figure(1)
plot(t,x,'-k',t,s,':k',t,p,'-.k',t,v,'--k'),
title('Initial trajectories of state variables');
ylabel('g/l');
xlabel('time in samples');
legend('biomass','substrate','product','volume','Location','NorthEast')

for k = 1:K
   % Calculation of the initial trajectories of the interconnections in time domain
    dx(k) = 10.0;
    ds(k) = 138;
    dp(k) = 25.0;
    dv(k) = 2.0;
end

T=16.0*ones(1,K);
pH=4.8*ones(1,K);
DO2=37*ones(1,K);
F=5.0*ones(1,K);

 %Calculation of the initial trajectories of Error_Lambda_v_(k)
 for k=1:K
    elx(k) = dx(k)-x(k+1);
    els(k) = ds(k)-s(k+1);
    elp(k) = dp(k)-p(k+1);
    elv(k) = dv(k)-v(k+1);
 end

% Start of the coordinating procedure
j=1
while j <= M2;%Second level iterations

%Calculation of the derivatives of the kinetic coefficients towards the control
variables
% Equations 5.33 to 5.35%

for k=1:K %Decomposition in time domain
    i=1
    while i <= M1 %First level iterations

    dmmaxdT(k) = a2+2*a3*T(k);
    dKsdT(k)=b2+2*b3*T(k);%Equation 5.33
    dYxsdT(k)=c2+2*c3*T(k);
    dYpxdT(k)=d2+2*d3*T(k);  % for the temperature

    dmmaxdpH(k)=a4+2*a5*pH(k);
```

```
    dKsdpH(k)=b4+2*b5*pH(k);
    dYxsdpH(k)=c4+2*c5*pH(k);
    dYpxdpH(k)=d4+2*d5*pH(k);  % for the pH

    dmmaxdDO2(k) =a6+2*a7*DO2(k);
    dKsdDO2(k) =b6+2*b7*DO2(k);
    dYxsdDO2(k) = c6+2*c7*DO2(k);
    dYpxdDO2(k) = d6+2*d7*DO2(k);  % for the DO2

    % Calculation of the kinetic coefficients

    mmax(k)=a1+a2*T(k)+a3*(T(k)^2)+a4*pH(k)+a5*(pH(k)^2)+a6*DO2(k)+a7*(DO2(k)^2);
    Ks(k)=b1+b2*T(k)+b3*(T(k)^2)+b4*pH(k)+b5*(pH(k)^2)+b6*DO2(k)+b7*(DO2(k)^2);
    Yxs(k)=c1+c2*T(k)+c3*(T(k)^2)+c4*pH(k)+c5*(pH(k)^2)+c6*DO2(k)+c7*(DO2(k)^2);
    Ypx(k)=d1+d2*T(k)+d3*(T(k)^2)+d4*pH(k)+d5*(pH(k)^2)+d6*DO2(k)+d7*(DO2(k)^2);

% Calculation of the expressions lambda_v_(k), k=0,K-1, v = x,s,p,v and lamda1(k),↵
lamda2(k)

pk=p(k);
ff(k)=dt*mmax(k) *((1-pk/pl)^0.52);%Equation 5.48
fx(k)=ff(k)*s(k)/(Ks(k) + s(k));
fs(k)=ff(k)*Ks(k) * x(k)/((Ks(k) + s(k))^2);%Equation 5.49
fp(k)=dt*mmax(k) * Ypx(k) *((1-p(k)/pl)^-0.48)*x(k)*s(k)/(pl*(Ks(k) + s(k)));%↵
Equation 5.51
fv(k)=dt*F(k)/(v(k)^2);%Equation 5.50

f1(k)=ff(k)*x(k)*s(k)/(Ks(k) + s(k));%Equation 5.52
f2(k)=f1(k)/(Ks(k) + s(k));%Equation 5.5

% Calculation of the expressions ee_v_(k)=lambda(k)-m_v*E_lambda_v_(k)

eex(k)=lx(k)-mx*elx(k);
ees(k)=ls(k)-ms*els(k);
eep(k)=lp(k)-mp*elp(k);
eev(k)=lv(k)-mv*elv(k);

%Calculation of the gradients for state variables
ex(k)=(1+fx(k)-dt*kd+dt*F(k)/v(k))*eex(k)+((fx(k)/Yxs(k))- dt*m)*ees(k)+dt*fx(k)*Ypx↵
(k) * eep(k);%Equation 5.54
es(k)=fs(k)*eex(k)+(1-fs(k)/Yxs(k) - dt*F(k)/v(k))*ees(k)+Ypx(k) * fs(k)*eep(k);%↵
Equation 5.55
ep(k)=(1+fp(k)-dt*F(k)/v(k))*eep(k);%Equation 5.56
ev(k)=fv(k)*x(k)*eex(k)-(fv(k)*(si-s(k))*ees(k))+(fv(k)*p(k)*eep(k))+ eev(k);%↵
Equation5.57

% Calculation of the gradients for control variables
eF(k) = dt*x(k)/v(k)*eex(k)+(dt*(si-s(k))/v(k)*ees(k))-(dt*p(k)/v(k)*eep(k))+dt*eev↵
(k);%Equation 5.58
eT(k) = (f1(k)*dmmaxdT(k)-f2(k)*dKsdT(k))*eex(k) + (-f1(k))*(dmmaxdT(k)*Yxs(k)-mmax↵
(k)*dYxsdT(k)/(Yxs(k)^2))+ ((f2(k)/Yxs(k))*dKsdT(k)*ees(k)) + (f1(k)*(dYpxdT(k)*mmax↵
(k)+Ypx(k)*dmmaxdT(k)))-(Ypx(k)*f2(k)*dKsdT(k))*eep(k);%Equation 5.59
epH(k) = (f1(k)*dmmaxdpH(k)-f2(k)*dKsdpH(k))*eex(k) + (-f1(k))*(dmmaxdpH(k)*Yxs(k)-↵
mmax(k)*dYxsdpH(k)/(Yxs(k)^2))+((f2(k)/Yxs(k))*dKsdpH(k)*ees(k))+(f1(k)*(dYpxdpH(k)↵
```

```
*mmax(k)+Ypx(k)*dmmaxdpH(k))-(Ypx(k)*f2(k)*dKsdpH(k))*eep(k));%Equation 5.60
eDO2(k) = (f1(k)*dmmaxdDO2(k)-f2(k)*dKsdDO2(k))*eex(k) + (-f1(k))*(dmmaxdDO2(k)*Yxs↙
(k)-mmax(k)*dYxsdDO2(k)/(Yxs(k)^2))+((f2(k)/Yxs(k))*dKsdT(k)*ees(k))+(f1(k)*↙
(dYpxdDO2(k)*mmax(k)+Ypx(k)*dmmaxdDO2(k))-(Ypx(k)*f2(k)*dKsdDO2(k))*eep(k));%↙
Equation 5.61
n1=[ex(k) es(k) ep(k) ev(k) eF(k) eT(k) epH(k) eDO2(k)]';

%Calculation norm of gradients

nn = norm(n1);
%Chech the condition for end of the iterations on the first level
if nn <=eps1
break
%Calculation of the new values of the state and control variables
elseif k == 1
        x(k)=x0;
        s(k)=s0;
        p(k)=p0;
        v(k)=v0;
        T(k) = T(k) + aT*eT(k);
        F(k) = F(k) + aF*eF(k);
        pH(k) = pH(k) + apH*epH(k);
        DO2(k) = DO2(k) + aDO2*eDO2(k);

    else
    x(k) = x(k) + ax*ex(k);
    s(k) = s(k) + as*es(k);
    p(k) = p(k) + ap*ep(k);
    v(k) = v(k) + av*ev(k);
    T(k) = T(k) + aT*eT(k);
    F(k) = F(k) + aF*eF(k);
    pH(k) = pH(k) + apH*epH(k);
    DO2(k) = DO2(k) + aDO2*eDO2(k);
end

%Check against constraints

if x(k) < xmin
    x(k) = xmin;
elseif x(k) > xmax
    x(k) = xmax;
end
if s(k) < smin
    s(k) = smin;
elseif s(k) > smax
    s(k) = smax;
end
if p(k) < pmin
    p(k) = pmin;
elseif p(k) > pmax
    p(k) = pmax;
end
if v(k) < vmin
    v(k) = vmin;
```

```
elseif v(k) > vmax
    v(k) = vmax;
end
if F(k) < Fmin
    F(k) = Fmin;
elseif F(k) > Fmax
    F(k) = Fmax;
end
if T(k) < Tmin
    T(k) = Tmin;
elseif T(k) > Tmax
   T(k) = Tmax;
end
if pH(k) < pHmin
    pH(k) = pHmin;
elseif pH(k) > pHmax
    pH(k) = pHmax;
end
if DO2(k) < DO2min
   DO2(k) = DO2min;
elseif DO2(k) > DO2max
    DO2(k) = DO2max;
end
i=i+1
    end %End of the iterations on the first level
end %end of the iteration for k=1,K

%Calculation of the final value of x(k)=x(K+1)=xK1;sK1,pK1,vK1

xK1=x(K)+f1(K)-dt*kd*x(K)-dt*F(K)*x(K)/v(K);
sK1=s(K)-f1(K)/Yxs(K)-dt*m*x(K)+dt*F(K)*(si-s(K))/v(K);
pK1=p(K)+f1(K)*Ypx(K)-dt*F(K)*p(K)/v(K);
vK1=v(K)+dt*F(K);
if xK1 <= xmin
    xK1 = xmin;
elseif xK1 >=xmax
    xK1 = xmax;
end


if sK1 <= smin
    sK1 = smin;
elseif sK1 >= smax
    sK1 = smax;
end


if pK1 <= pmin
    pK1 = pmin;
elseif pK1 >= pmax
    pK1=pmax
end
```

```
if vK1 <= vmin
    vK1 = vmin;
elseif vK1 >= vmax
    vK1 = vmax;
end
x(K+1)=xK1
p(K+1)=pK1
s(K+1)=sK1
v(K+1)=vK1

for k=1:K
    elx(k) = dx(k) - x(k+1);%Equation 5.25
    els(k) = ds(k) - s(k+1);%Equation 5.26
    elp(k) = dp(k) - p(k+1);%Equation 5.27
end
nn1=norm(elx)+ norm(els)+ norm(elp)+ norm(elv);

if nn1<=eps2
    break

else j=j+1;
end
for k =1:K
    lx(k) = lx(k) - alx*elx(k);
    ls(k) = ls(k) - als*els(k);
    lp(k) = lp(k) - alp*elp(k);
    lv(k) = lv(k) - alv*elv(k);

    dx(k) = dx(k) + adx*mx*elx(k);
    ds(k) = ds(k) + ads*ms*els(k);
    dp(k) = dp(k) + adp*mp*elp(k);
    dv(k) = dv(k) + adv*mv*elv(k);

end
end

ti=toc;

%Plot Results
figure(2)
t=1:K+1
plot(t,x,'-k',t,p,':k',t,v,'-.k')
title('Optimised Results for the biomass, product and volume states')
ylabel('g/l');
xlabel('time in samples');
legend('biomass','product','volume','Location','NorthWest')

figure(3)
title('Optimised Results for the substrate')
ylabel('g/l');
xlabel('time in samples');
plot(s,'-k')
legend('substrate','Location','NorthEast')
t1=1:K
```

```
figure(4)
plot(t1,T,'-k',t1,pH,':k',t1,DO2,'-.k',t1,F,'--k')
title('Optimised Results for the Temperature T, pH, DO2,input flow rate F')
ylabel('g/l');
xlabel('time in samples');
legend('Temperature T', 'pH', 'DO2','Input flow rate
F','Location','NorthEastOutside')

figure(5)
plot(t1,lx,'-k',t1,ls,':k',t1,lp,'-.k',t1,lv,'--k')
title('Optimised Results for the conjugate variables lambda')
ylabel('g/l');
xlabel('time in samples');
legend('Lambda x','Lambda s','Lambda p','Lambda v','Location','SouthWest')

%Experimental results from the Department of Chemical Enginering without
optimization
xe =
[0.83,0.85,0.86,0.88,0.9,0.91,0.92,0.92,0.93,1,1.13,1.2,1.33,1.44,1.55,1.66,1.77,2.2
2,2.65,3.21,3.53,4.09,4.52,5.02,5.5,6.13,6.92,7.74,8.33,9.05,9.75,10.45,11.17,11.86,
12.55,13.22,13.93,14.02,14.02,14.05,14.1,14.16,14.2,14.3,14.3,14.32,14.35,14.39,14.4
];
se =
[154,153.3,152.27,151.9,151.33,150.7,150,149.3,148.67,147.7,146.83,145.6,145,143.8,1
42.67,141.5,140.33,138.3,136,133.8,131.67,129.1,127,124.7,122.33,120.2,118.17,116.1,
114,112.2,109.33,106.9,104.67,101.5,98.5,95.4,92.33,92.8,93.42,94,94.5,94.4,94.35,94
.3,94.2,94.3,94.4,94.5,94.6];
pe =
[2.1,2.18,2.28,2.36,2.45,2.53,2.63,2.71,2.8,3.15,3.5,3.85,4.2,4.55,4.9,5.25,5.6,6.78
,7.99,9.16,10.38,11.51,12.76,13.97,15.15,16.67,18.21,19.98,21.28,22.8,24.34,25.84,27
.4,28.73,30.05,30,29.03,28.74,28.52,28.26,28,27.7,27.5,27.3,27,26.8,26.5,26.3,26]
k = 1:K+1;
figure(6)
plot (k,xe,'-k',k,x,'--k')
legend('Experiment biomass','Optimal biomass','Location','NorthWest')
title('Comparison between experimental and Optimal trajectories')
ylabel('g/l');
xlabel('time in samples');
hold off
figure(7)
plot (k,se,'-k',k,s,'--k')
title('Comparison between experimental and Optimal trajectories')
legend('Experiment substrate','Optimal substrate','Location','NorthEast')
ylabel('g/l');
xlabel('time in samples');
hold off
figure(8)
plot (k,pe,'-k',k,p,'--k')
ylabel('g/l');
xlabel('time in samples');
title('Comparison between experimental and Optimal trajectories')
legend('Experiment product','Optimal product','Location','NorthWest')
hold off
figure(9)
```

```
plot(k,xe,'-k',k,se,'--k',k,pe,'-.k')
ylabel('g/l');
xlabel('time in samples');
title('Experimental tragectories')
legend('Biomass','Substrate','Product', 'Location','NorthEast')
```

```
%Create function
function ydot=bioreactorfc(t,y)
global par T pH DO2 F kd m pl n si
%Model_variables;
ydot(1) =(par(1)+par(2)*T+par(3)*(T*T)+par(4)*pH+par(5)*(pH*pH)+par(6)*DO2+par(7)*(DO2*DO2))*((1-(y(3)/pl))^n)*y(1)*y(2)/↵
((par(8)+par(9)*T+par(10)*(T*T)+par(11)*pH+par(12)*(pH*pH)+par(13)*DO2+par(14)*(DO2*DO2))+y(2))-(kd*y(1))-F*y(1)/y(4);
ydot(2) =(par(1)+par(2)*T+par(3)*(T*T)+par(4)*pH+par(5)*(pH*pH)+par(6)*DO2+par(7)*(DO2*DO2))*((1-(y(3)/pl))^n)*y(1)*y(2)/↵
((par(8)+par(9)*T+par(10)*(T*T)+par(11)*pH+par(12)*(pH*pH)+par(13)*DO2+par(14)*(DO2*DO2))+y(2))/-(par(15)+par(16)*T+par(17)*↵
(T*T)+par(18)*pH+par(19)*(pH*pH)+par(20)*DO2+par(21)*(DO2*DO2))-m*y(1)-F*(si-y(2))/y(4);
ydot(3) =(par(1)+par(2)*T+par(3)*(T*T)+par(4)*pH+par(5)*(pH*pH)+par(6)*DO2+par(7)*(DO2*DO2))*((1-(y(3)/pl))^n)*y(1)*y(2)/↵
((par(8)+par(9)*T+par(10)*(T*T)+par(11)*pH+par(12)*(pH*pH)+par(13)*DO2+par(14)*(DO2*DO2))+y(2))*(par(22)+par(23)*T+par(24)*↵
(T*T)+par(25)*pH+par(26)*(pH*pH)+par(27)*DO2+par(28)*(DO2*DO2))-F*y(3)/y(4);
ydot(4)=F
ydot=[ydot(1) ydot(2) ydot(3) ydot(4)]';
```

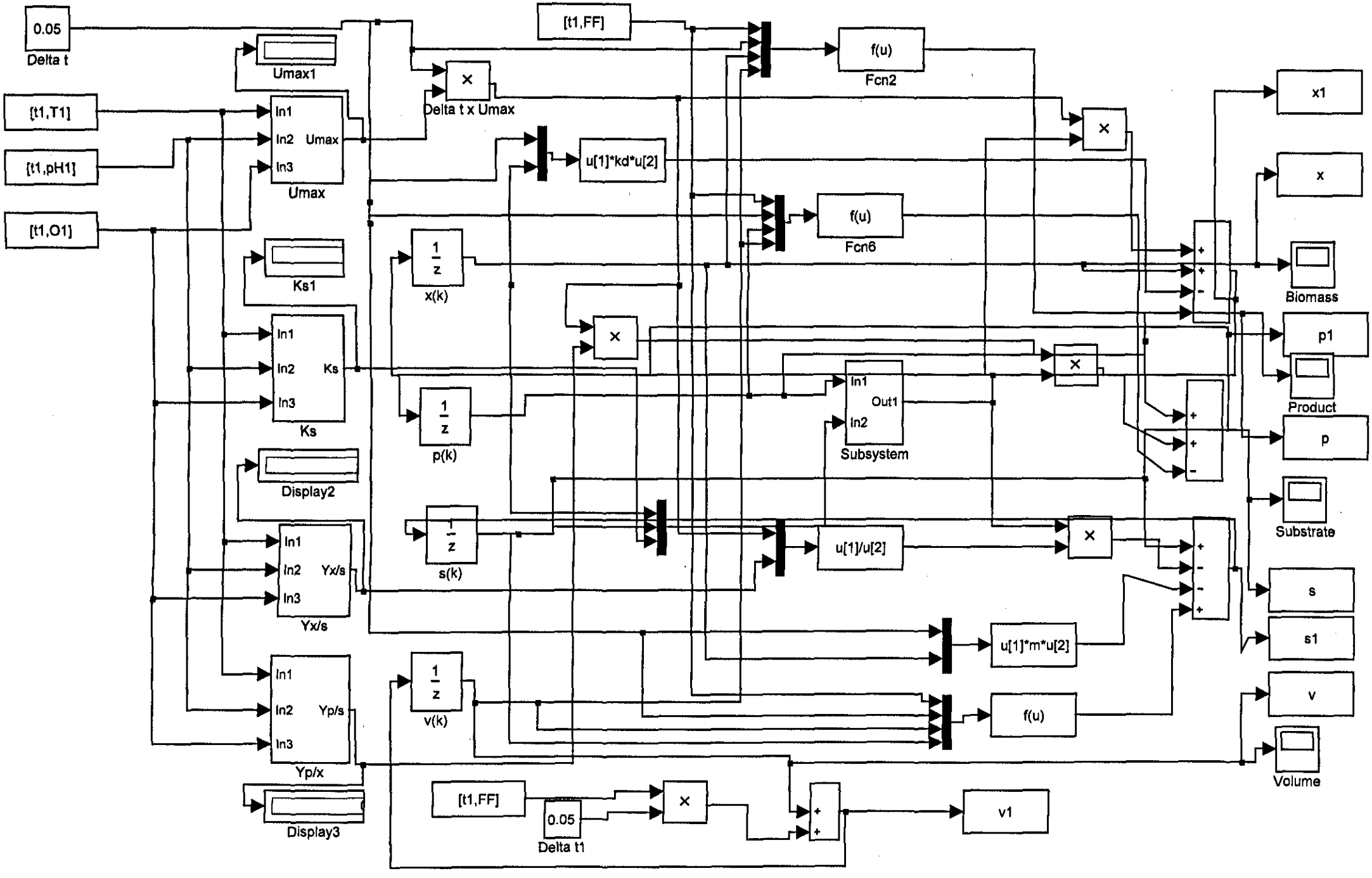
Appendix B

1. Simulink program Opt_Model.mdl for simulation of the fed-batch fermentation process.

(1 page)

2. Matlab script YM_var.m for introduction of the simulation parameters.

(2 pages)


0.05
Delta t
Umax1
[t1,T1]
[t1,pH1]
[t1,O1]
In1
In2 Umax
In3
Umax
Ks1
In1
In2 Ks
In3
Ks
Display2
In1
In2 Yx/s
In3
Yx/s
In1
In2 Yp/s
In3
Yp/x
Display3
[t1,FF]
Delta t x Umax
u[1]*kd*u[2]
1/z
x(k)
1/z
p(k)
1/z
s(k)
1/z
v(k)
[t1,FF]
0.05
Delta t1
f(u)
Fcn2
f(u)
Fcn6
In1
Out1
In2
Subsystem
u[1]/u[2]
u[1]*m*u[2]
f(u)
x1
x
Biomass
p1
Product
p
Substrate
s
s1
v
Volume
v1

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Coefficients for Simulink simulation       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TT = 48;
t  = 1:1:TT;
F = 0;
si= 266.6;
pH = 4.8;
T  = 16;
O  = 37;
pH = pH*ones(1,TT);
T  = T*ones(1,TT);
O  = O*ones(1,TT);
F = F*ones(1,TT);
a1 = 0.0004;
a2 = 0.0001;
a3 = 0.0002;
a4 = 0.0001;
a5 = 0.0001;
a6 = 0.00002;
a7 = 0.00002;

b1 = 4;
b2 = 0.5;
b3 = 0.13;
b4 = 0.13;
b5 = 0.013;
b6 = 0.014;
b7 = 0.013;

c1 = 0.0002;
c2 = 0.0001;
c3 = 0.000021;
c4 = 0.00021;
c5 = 0.000211;
c6 = 0.0000211;
c7 = 0.0002;

d1 = 0.0360;
d2 = 0.008;
d3 = 0.003;
d4 = 0.005;
d5 = 0.005;
d6 = 0.001;
d7 = 0.0002;

x0 = 1.83;
s0 = 150.0;
p0 = 2.1;
v0 = 2.1;

t1=t';
pH1=pH';
T1=T';
```

```
O1=O';
FF=F';
k = 0.00008;
pl = 50;
m  = 0.11;
n  = 0.52;

sim('quadratic_model')
figure (1)
plot(x)
hold off
figure (2)
plot(s)
hold off
figure (3)
plot(p)
hold off
figure (4)
plot(v)
hold off
```

Appendix C

1. Program optconfer1.m for sequential implementation of the parallel algorithm.

(6 pages)

2. Program subprob1.m for sequential implementation of the parallel algorithm.

(5 pages)

3. Program optconfer2.m for parallel implementation of the parallel algorithm.

(6 pages)

4. Program subprob2.m for parallel implementation of the parallel algorithm.

(5 pages)

```
clear all
%close all

global par T pH DO2 F kd m pl n si x1 s1 p1 v1 T1 pH1 DO21 F1

%K M1 eps1 as ac mu l el conmin conmax statemin statemax
%Set Initial values
M1 = 5; M2 = 5;%Max number of iterations of the first and second  levels
K = 48;%Number of steps in the optimization horizon

ax = 0.001;as = 0.001; ap = 0.001;av = 0.001;%Steps of the gradient procedures for calculation of the State variables
aT = 0.05; aF = 0.05; apH = 0.05; aDO2 = 0.05;%Steps of the gradient procedures for calculation of the Control variables
alx = 0.001; als = 0.001; alp = 0.001; alv = 0.001;%Step of the gradient procedures for calculation of the Conjugate variables
adx = 0.001; ads = 0.001; adp = 0.001; adv = 0.001;%Steps of the gradient procedures for calculation of the Interconnections in time

eps1=0.001;
eps2=0.001;%tolerances for stpo of the calculation of the first and second levels
%epsx = 0.01; epss = 0.01; epsp = 0.01; epsv = 0.01;%Conjugate error variables
%epsF = 0.01; epsT = 0.01; epspH = 0.01; epsDO2 = 0.01;%Control error variables

%Initial values of the penalty coefficients
mx=0.001;
ms=0.001;
mp=0.001;
mv=0.001;
astate=[ax as ap av];
acon=[aT aF apH aDO2];
al=[alx als alp alv];
mu=[mx ms mp mv];
%Initial trajectory values of the conjugate and control variables

for k = 1:K
 %Initial trajectories for the conjugate variables
        lx(k)=0.001;
        ls(k)=0.005;
        lp(k)=0.002;
        lv(k)=0.003;
end
l=[lx;ls;lp;lv];

%Initial trajectories of the control variables
T=16;
pH=4.8;
DO2=37;
F=0.;


% Model co-efficients
a1 = 0.0004; a2 = 0.0001;a3 = 0.0002; a4 = 0.0001; a5 = 0.0001; a6 = 0.00002; a7 = 0.00002;
```

```
%b1 = 0.2; b2 = 0.03;b3 = 0.01; b4 = 0.01; b5 = 0.001; b6 = 0.002; b7 = 0.001;
b1 = 4; b2 = 0.5 ;b3 = 0.13; b4 = 0.13; b5 = 0.013; b6 = 0.014; b7 = 0.013;
%c1 = 2; c2 = 0.3; c3 = 0.021; c4 = 0.051; c5 = 0.00211; c6 = 0.0211; c7 = 0.002;
c1 = 0.0002; c2 = 0.0001; c3 = 0.000021; c4 = 0.00021; c5 = 0.000211; c6 =
0.0000211; c7 = 0.0002;
%d1 = 0.0040; d2 = 0.006; d3 = 0.001; d4 = 0.005; d5 = 0.0005; d6 = 0.001; d7 =
0.0002;
d1 = 0.0360; d2 = 0.008; d3 = 0.003; d4 = 0.005; d5 = 0.005; d6 = 0.001; d7 =
0.0002;
par = [a1 a2 a3 a4 a5 a6 a7 b1 b2 b3 b4 b5 b6 b7 c1 c2 c3 c4 c5 c6 c7 d1 d2 d3 d4 d5
d6 d7];

kd = 0.00008;
pl = 50;
m  = 0.11;
n  = 0.52;
si= 266.6;
dt = 2.5;%Delta t

%Initial values of state variables
x0=1.83;
s0=150.0;
p0=2.1;
v0=2.1;
y0=[x0 s0 p0 v0];

%Constraints of state and control variables
Tmin = 12;   %%%%%%%%%%%%%%%%%%%%%%%%
pHmin = 4;  %                        %
DO2min = 10; %                        %
Fmin = 0;   % For control variables%
Tmax = 20; %                        %
pHmax = 7; %                        %
DO2max = 60;%                        %
Fmax = 5.0;  %%%%%%%%%%%%%%%%%%%%%%%%
% State variables%
vmin = 2;
vmax = 5;
pmin = 2.0;
pmax = 70;
smin = 20;
smax = 300;
xmin = 0.83;
xmax =55;

conmin=[Tmin pHmin DO2min Fmin];
conmax=[Tmax pHmax DO2max Fmax];
statemin=[xmin smin pmin vmin];
statemax=[xmax smax pmax vmax];

%Calculation of the initial trajectories of state variables
t=1:K+1
[t,y]=ode15s('model_func1',t,y0)
x=(y(:,1))';
```

```
s=(y(:,2))';
p=(y(:,3))';
v=(y(:,4))'; %transformation to raw vectors

figure(1)
plot(t,x,'-k',t,s,':k',t,p,'-.k',t,v,'--k'),
title('Initial trajectories of state variables');
ylabel('g/l');
xlabel('time in samples');
legend('biomass','substrate','product','volume','Location','NorthEast')



for k = 1:K
   % Calculation of the initial trajectories of the interconnections in time domain
    dx(k) = 10.;
    ds(k) = 138.0;
    dp(k) = 25.0;
    dv(k) = 2.0;
end

T=16.0*ones(1,K);
pH=4.8*ones(1,K);
DO2=37*ones(1,K);
F=5.0*ones(1,K);

 %Calculation of the initial trajectories of Error_Lambda_v_(k)
 for k=1:K
    elx(k) = dx(k)-x(k+1);
    els(k) = ds(k)-s(k+1);
    elp(k) = dp(k)-p(k+1);
    elv(k) = dv(k)-v(k+1);
 end
 el=[elx;els;elp;elv];



%The function subprog.m is on the client path, but it has to be availlable
%to the workers.One way to do this is to use the FileDepandencies property
%of the pjob.

% Start of the coordinating procedure
j=1
while j <= M2;%Second level iterations
 %jm=findResource('scheduler','type','jobmanager',...
   % 'name','Myjobmanager1','LookupURL','TzonevaRa')
%pjob=createParallelJob(jm)
%set(pjob,'MinimumNumberOfWorkers',8)
%set(pjob,'MaximumNumberOfWorkers',8)
%set(pjob,'FileDependencies',{'subprob.m'})
%Starting parallel computing

%Create task- only one with 8 output arguments  and not input arguments
%task=createTask(pjob, @subprob,8,{})
```

```
%submit(pjob)

%wait for the end of calculation
%waitForState(pjob,'finished')
%results=getAllOutputArguments(pjob)
%get(pjob,'StartTime')
%get(pjob,'FinishTime')
%Calculation of the improved values of the coordinating variables
%Calculation of the errors e_lambda_V (k)
numlabs=8;
dd=K/numlabs;
for labindex=numlabs:-1:1
    a=(labindex-1)*dd+1
    b=labindex*dd
   [x,s,p,v,T,pH,DO2,F]=subprob1(a,b,par,T,pH,DO2,F,kd,m,pl,n,si,K,M1,eps1,mu,l,x,s,↙
p,v,el,conmin,conmax,statemin,statemax,astate,acon,labindex,dt,y0,numlabs)
end

for k=1:K
    elx(k) = dx(k) - x(k+1);%Equation 5.25
    els(k) = ds(k) - s(k+1);%Equation 5.26
    elp(k) = dp(k) - p(k+1);%Equation 5.27
    elv(k) = dv(k) - v(k+1);%Equation 5.28
end
el=[elx;els;elp;elv];
%Calculation of the norm of the errors
nn1=norm(elx)+ norm(els)+ norm(elp)+ norm(elv);
%Check of the convergency of the coordinating procedure for the end of the
%calculations
if nn1<=eps2
    break

else

for k =1:K
    lx(k) = lx(k) - alx*elx(k);
    ls(k) = ls(k) - als*els(k);
    lp(k) = lp(k) - alp*elp(k);
    lv(k) = lv(k) - alv*elv(k);

    dx(k) = dx(k) + adx*mx*elx(k);
    ds(k) = ds(k) + ads*ms*els(k);
    dp(k) = dp(k) + adp*mp*elp(k);
    dv(k) = dv(k) + adv*mv*elv(k);
end
end
l=[lx;ls;lp;lv];
%destroy(pjob)
j=j+1
end
%Plot Results
figure(2)
t=1:K+1
plot(t,x,'-k',t,p,':k',t,v,'-.k')
```

```
title('Optimised Results for the biomass, product and volume states')
ylabel('g/l');
xlabel('time in samples');
legend('biomass','product','volume','Location','NorthWest')

figure(3)
title('Optimised Results for the substrate')
ylabel('g/l');
xlabel('time in samples');
plot(s,'-k')
legend('substrate','Location','NorthEast')
t1=1:K
figure(4)
plot(t1,T,'-k',t1,pH,':k',t1,DO2,'-.k',t1,F,'--k')
title('Optimised Results for the Temperature T, pH, DO2,input flow rate F')
ylabel('g/l');
xlabel('time in samples');
legend('Temperature T', 'pH', 'DO2','Input flow rate↙
F','Location','NorthEastOutside')

figure(5)
plot(t1,lx,'-k',t1,ls,':k',t1,lp,'-.k',t1,lv,'--k')
title('Optimised Results for the conjugate variables lambda')
ylabel('g/l');
xlabel('time in samples');
legend('Lambda x','Lambda s','Lambda p','Lambda v','Location','SouthWest')

%Experimental results from the Department of Chemical Enginering without↙
optimization
xe =↙
[0.83,0.85,0.86,0.88,0.9,0.91,0.92,0.92,0.93,1,1.13,1.2,1.33,1.44,1.55,1.66,1.77,2.2↙
2,2.65,3.21,3.53,4.09,4.52,5.02,5.5,6.13,6.92,7.74,8.33,9.05,9.75,10.45,11.17,11.86,↙
12.55,13.22,13.93,14.02,14.02,14.05,14.1,14.16,14.2,14.3,14.3,14.32,14.35,14.39,14.4↙
];
se =↙
[154,153.3,152.27,151.9,151.33,150.7,150,149.3,148.67,147.7,146.83,145.6,145,143.8,1↙
42.67,141.5,140.33,138.3,136,133.8,131.67,129.1,127,124.7,122.33,120.2,118.17,116.1,↙
114,112.2,109.33,106.9,104.67,101.5,98.5,95.4,92.33,92.8,93.42,94,94.5,94.4,94.35,94↙
.3,94.2,94.3,94.4,94.5,94.6];
pe =↙
[2.1,2.18,2.28,2.36,2.45,2.53,2.63,2.71,2.8,3.15,3.5,3.85,4.2,4.55,4.9,5.25,5.6,6.78↙
,7.99,9.16,10.38,11.51,12.76,13.97,15.15,16.67,18.21,19.98,21.28,22.8,24.34,25.84,27↙
.4,28.73,30.05,30,29.03,28.74,28.52,28.26,28,27.7,27.5,27.3,27,26.8,26.5,26.3,26]
k = 1:K+1;
figure(6)
plot (k,xe,'-k',k,x,'--k')
legend('Experiment biomass','Optimal biomass','Location','NorthWest')
title('Comparison between experimental and Optimal trajectories')
ylabel('g/l');
xlabel('time in samples');
hold off
figure(7)
plot (k,se,'-k',k,s,'--k')
title('Comparison between experimental and Optimal trajectories')
```

```
legend('Experiment substrate','Optimal substrate','Location','NorthEast')
ylabel('g/l');
xlabel('time in samples');
hold off
figure(8)
plot (k,pe,'-k',k,p,'--k')
ylabel('g/l');
xlabel('time in samples');
title('Comparison between experimental and Optimal trajectories')
legend('Experiment product','Optimal product','Location','NorthWest')
hold off
figure(9)
plot(k,xe,'-k',k,se,'--k',k,pe,'-.k')
ylabel('g/l');
xlabel('time in samples');
title('Experimental tragectories')
legend('Biomass','Substrate','Product', 'Location','NorthEast')
```

```
function [x,s,p,v,T,pH,DO2,F]=subprob1(a,b,par,T,pH,DO2,F,kd,m,pl,n,si,K,M1,eps1,mu,l,x,s,p,v,el,conmin,conmax,statemin,statemax,astate,acon,labindex,dt,y0,numlabs)
global par T pH DO2 F kd m pl n si x1 s1 p1 v1 T1 pH1 DO21 F1
%Calculation of the initial and final point of the subintervals
%dd=K/numlabs
%a=(labindex-1)*dd+1
%b=labindex*dd

for k=a:b %Decomposition in time domain
    i=1
    while i <= M1 %First level iterations

    dmmaxdT(k) = par(2)+2*par(3)*T(k);
    dKsdT(k)=par(9)+2*par(10)*T(k);%Equation 5.33
    dYxsdT(k)=par(16)+2*par(17)*T(k);
    dYpxdT(k)=par(23)+2*par(24)*T(k);  % for the temperature

    dmmaxdpH(k)=par(4)+2*par(5)*pH(k);
    dKsdpH(k)=par(11)+2*par(12)*pH(k);
    dYxsdpH(k)=par(18)+2*par(19)*pH(k);
    dYpxdpH(k)=par(25)+2*par(26)*pH(k);  % for the pH

    dmmaxdDO2(k) =par(6)+2*par(7)*DO2(k);
    dKsdDO2(k) =par(13)+2*par(14)*DO2(k);
    dYxsdDO2(k) = par(20)+2*par(21)*DO2(k);
    dYpxdDO2(k) = par(27)+2*par(28)*DO2(k);  % for the DO2

    % Calculation of the kinetic coefficients

    mmax(k)=par(1)+par(2)*T(k)+par(3)*(T(k)^2)+par(4)*pH(k)+par(5)*(pH(k)^2)+par(6)*DO2(k)+par(7)*(DO2(k)^2);
    Ks(k)=par(8)+par(9)*T(k)+par(10)*(T(k)^2)+par(11)*pH(k)+par(12)*(pH(k)^2)+par(13)*DO2(k)+par(14)*(DO2(k)^2);
    Yxs(k)=par(15)+par(16)*T(k)+par(17)*(T(k)^2)+par(18)*pH(k)+par(19)*(pH(k)^2)+par(20)*DO2(k)+par(21)*(DO2(k)^2);
    Ypx(k)=par(22)+par(23)*T(k)+par(24)*(T(k)^2)+par(25)*pH(k)+par(26)*(pH(k)^2)+par(27)*DO2(k)+par(28)*(DO2(k)^2);

% Calculation of the expressions lambda_v_(k), k=0,K-1, v = x,s,p,v and lamda1(k), lamda2(k)


ff(k)=dt*mmax(k) *((1-p(k)/pl)^0.52);%Equation 5.48
fx(k)=ff(k)*s(k)/(Ks(k) + s(k));
fs(k)=ff(k)*Ks(k) * x(k)/((Ks(k) + s(k))^2);%Equation 5.49
fp(k)=dt*mmax(k) * Ypx(k) *((1-p(k)/pl)^-0.48)*x(k)*s(k)/(pl*(Ks(k) + s(k)));%Equation 5.51
fv(k)=dt*F(k)/(v(k)^2);%Equation 5.50

f1(k)=ff(k)*x(k)*s(k)/(Ks(k) + s(k));%Equation 5.52
f2(k)=f1(k)/(Ks(k) + s(k));%Equation 5.5

% Calculation of the expressions ee_v_(k)=lambda(k)-m_v*E_lambda_v_(k)
```

```
eex(k)=l(1,k)-mu(1)*el(1,k);
ees(k)=l(2,k)-mu(2)*el(2,k);
eep(k)=l(3,k)-mu(3)*el(3,k);
eev(k)=l(4,k)-mu(4)*el(4,k);

%Calculation of the gradients for state variables
ex(k)=(1+fx(k)-dt*kd+dt*F(k)/v(k))*eex(k)+((fx(k)/Yxs(k))- dt*m)*ees(k)+dt*fx(k)*Ypx
(k) * eep(k);%Equation 5.54
es(k)=fs(k)*eex(k)+(1-fs(k)/Yxs(k) - dt*F(k)/v(k))*ees(k)+Ypx(k) * fs(k)*eep(k);%
Equation 5.55
ep(k)=(1+fp(k)-dt*F(k)/v(k))*eep(k);%Equation 5.56
ev(k)=fv(k)*x(k)*eex(k)-(fv(k)*(si-s(k))*ees(k))+(fv(k)*p(k)*eep(k))+ eev(k);%
Equation5.57

% Calculation of the gradients for control variables
eF(k) = dt*x(k)/v(k)*eex(k)+(dt*(si-s(k))/v(k)*ees(k))-(dt*p(k)/v(k)*eep(k))+dt*eev
(k);%Equation 5.58
eT(k) = (f1(k)*dmmaxdT(k)-f2(k)*dKsdT(k))*eex(k) + (-f1(k))*(dmmaxdT(k)*Yxs(k)-mmax
(k)*dYxsdT(k)/(Yxs(k)^2))+ ((f2(k)/Yxs(k))*dKsdT(k)*ees(k)) + (f1(k)*(dYpxdT(k)*mmax
(k)+Ypx(k)*dmmaxdT(k)))-(Ypx(k)*f2(k)*dKsdT(k))*eep(k);%Equation 5.59
epH(k) = (f1(k)*dmmaxdpH(k)-f2(k)*dKsdpH(k))*eex(k) + (-f1(k))*(dmmaxdpH(k)*Yxs(k)-
mmax(k)*dYxsdpH(k)/(Yxs(k)^2))+((f2(k)/Yxs(k))*dKsdpH(k)*ees(k))+(f1(k)*(dYpxdpH(k)
*mmax(k)+Ypx(k)*dmmaxdpH(k))-(Ypx(k)*f2(k)*dKsdpH(k))*eep(k));%Equation 5.60
eDO2(k) = (f1(k)*dmmaxdDO2(k)-f2(k)*dKsdDO2(k))*eex(k) + (-f1(k))*(dmmaxdDO2(k)*Yxs
(k)-mmax(k)*dYxsdDO2(k)/(Yxs(k)^2))+((f2(k)/Yxs(k))*dKsdT(k)*ees(k))+(f1(k)*
(dYpxdDO2(k)*mmax(k)+Ypx(k)*dmmaxdDO2(k))-(Ypx(k)*f2(k)*dKsdDO2(k))*eep(k));%
Equation 5.61
n1=[ex(k) es(k) ep(k) ev(k) eF(k) eT(k) epH(k) eDO2(k)]';

%Calculation norm of gradients

nn = norm(n1)
%Check the condition for end of the iterations on the first level
if nn <=eps1
break
%Calculation of the new values of the state and control variables
elseif k == 1
        x1(1,k,labindex)=y0(1);
        s1(1,k,labindex)=y0(2);
        p1(1,k,labindex)=y0(3);
        v1(1,k,labindex)=y0(4);
        T1(1,k,labindex) = T(k) + acon(1)*eT(k)
        pH1(1,k,labindex) = pH(k) + acon(2)*epH(k);
        DO21(1,k,labindex) = DO2(k) + acon(3)*eDO2(k);
        F1(1,k,labindex) = F(k) + acon(4)*eF(k);

else
    x1(1,k,labindex) = x(k) + astate(1)*ex(k);
    s1(1,k,labindex) = s(k) + astate(2)*es(k);
    p1(1,k,labindex) = p(k) + astate(3)*ep(k);
    v1(1,k,labindex) = v(k) + astate(4)*ev(k);
    T1(1,k,labindex) = T(k) + acon(1)*eT(k)
    pH1(1,k,labindex) = pH(k) + acon(2)*epH(k);
    DO21(1,k,labindex) = DO2(k) + acon(3)*eDO2(k);
```

```
    F1(1,k,labindex) = F(k) + acon(4)*eF(k);
    end
i=i+1
end

%Check against constraints

if x1(1,k,labindex) < statemin(1)
    x1(1,k,labindex) = statemin(1);
elseif x1(1,k,labindex) > statemax(1)
    x1(1,k,labindex) = statemax(1);
end
if s1(1,k,labindex) <statemin(2)
    s1(1,k,labindex) = statemin(2);
elseif s1(1,k,labindex) > statemax(2)
    s1(1,k,labindex) = statemax(2);
end
if p1(1,k,labindex) < statemin(3)
    p1(1,k,labindex) = statemin(3);
elseif p1(1,k,labindex) > statemax(3)
    p1(1,k,labindex) = statemax(3);
end
if v1(1,k,labindex) < statemin(4)
    v1(1,k,labindex) = statemin(4);
elseif v1(1,k,labindex) > statemax(4)
    v1(1,k,labindex) = statemax(4);
end

if T1(1,k,labindex) < conmin(1)
    T1(1,k,labindex) = conmin(1);
elseif T1(1,k,labindex) > conmax(1)
   T1(1,k,labindex) = conmax(1);
end
if pH1(1,k,labindex) < conmin(2)
    pH1(1,k,labindex) = conmin(2);
elseif pH1(1,k,labindex) > conmax(2)
    pH1(1,k,labindex) = conmax(2);
end
if DO21(1,k,labindex) < conmin(3)
   DO21(1,k,labindex) = conmin(3);
elseif DO21(1,k,labindex) > conmax(3)
    DO21(1,k,labindex) = conmax(3);
end
if F1(1,k,labindex) < conmin(4)
    F1(1,k,labindex) = conmin(4);
elseif F1(1,k,labindex) > conmax(4)
    F1(1,k,labindex) = conmax(4);
end


end  %end of the iterations in the subproblem time interval (a,b)

%labBarrier
dd=K/numlabs
```

```
if labindex==1

    x2=reshape(x1,1,prod(size(x1)));
    s2=reshape(s1,1,prod(size(s1)));
    p2=reshape(p1,1,prod(size(p1)));
    v2=reshape(v1,1,prod(size(v1)));
    T2=reshape(T1,1,prod(size(T1)))
    pH2=reshape(pH1,1,prod(size(pH1)));
    DO22=reshape(DO21,1,prod(size(DO21)));
    F2=reshape(F1,1,prod(size(F1)));
    x=x2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):(3*K+b+3*dd)↙
(4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)↙
(7*K+a+7*dd):(7*K+b+7*dd)])
    s=s2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):(3*K+b+3*dd)↙
(4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)↙
(7*K+a+7*dd):(7*K+b+7*dd)])
    p=p2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):(3*K+b+3*dd)↙
(4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)↙
(7*K+a+7*dd):(7*K+b+7*dd)])
    v=v2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):(3*K+b+3*dd)↙
(4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)↙
(7*K+a+7*dd):(7*K+b+7*dd)])
    T=T2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):(3*K+b+3*dd)↙
(4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)↙
(7*K+a+7*dd):(7*K+b+7*dd)])
    pH=pH2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):↙
(3*K+b+3*dd) (4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):↙
(6*K+b+6*dd) (7*K+a+7*dd):(7*K+b+7*dd)])
    DO2=DO22([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):↙
(3*K+b+3*dd) (4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):↙
(6*K+b+6*dd) (7*K+a+7*dd):(7*K+b+7*dd)])
    F=F2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):(3*K+b+3*dd)↙
(4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)↙
(7*K+a+7*dd):(7*K+b+7*dd)])

        %Calculation of the final value of x(k)=x(K+1)=xK1;sK1,pK1,vK1
    mmax(K)=par(1)+par(2)*T(K)+par(3)*(T(K)^2)+par(4)*pH(K)+par(5)*(pH(K)^2)+par(6)↙
*DO2(K)+par(7)*(DO2(K)^2)
    Ks(K)=par(8)+par(9)*T(K)+par(10)*(T(K)^2)+par(11)*pH(K)+par(12)*(pH(K)^2)+par↙
(13)*DO2(K)+par(14)*(DO2(K)^2)
    Yxs(K)=par(15)+par(16)*T(K)+par(17)*(T(K)^2)+par(18)*pH(K)+par(19)*(pH(K)^2)+par↙
(20)*DO2(K)+par(21)*(DO2(K)^2)
    Ypx(K)=par(22)+par(23)*T(K)+par(24)*(T(K)^2)+par(25)*pH(K)+par(26)*(pH(K)^2)+par↙
(27)*DO2(K)+par(28)*(DO2(K)^2)
    ff(K)=dt*mmax(K) *((1-p(K)/pl)^0.52)
    f1(K)=ff(K)*x(K)*s(K)/(Ks(K) + s(K))
    xK1=x(K)+f1(K)-dt*kd*x(K)-dt*F(K)*x(K)/v(K);
    sK1=s(K)-f1(K)/Yxs(K)-dt*m*x(K)+dt*F(K)*(si-s(K))/v(K);
    pK1=p(K)+f1(K)*Ypx(K)-dt*F(K)*p(K)/v(K);
    vK1=v(K)+dt*F(K);
    if xK1 <= statemin(1)
        xK1 = statemin(1);
        elseif xK1 >=statemax(1)
    xK1 = statemax(1);
```

```
    end
x(49)=xK1

if sK1 <= statemin(2)
    sK1 = statemin(2);
elseif sK1 >= statemax(2)
    sK1 = statemax(2);
end
s(49)=sK1

if pK1 <= statemin(3)
    pK1 = statemin(3);
elseif pK1 >= statemax(3)
    pK1=statemax(3)
end
 p(49)=pK1

if vK1 <= statemin(4)
    vK1 = statemin(4);
elseif vK1 >= statemax(4)
    vK1 = statemax(4);
end
v(49)=vK1
end
```

```
clear all
%close all
global par T pH DO2 F kd m pl n si  K M1 eps1 mu l x s p v el conmin conmax statemin
statemax astate acon dt y0 x1 s1 p1 v1 T1 pH1 DO21 F1

%Set Initial values
M1 = 5; M2 = 5;%Max number of iterations of the first and second  levels
K = 48;%Number of steps in the optimization horizon

ax = 0.001;as = 0.001; ap = 0.001;av = 0.001;%Steps of the gradient procedures for
calculation of the State variables
aT = 0.05; aF = 0.05; apH = 0.05; aDO2 = 0.05;%Steps of the gradient procedures for
calculation of the Control variables
alx = 0.001; als = 0.001; alp = 0.001; alv = 0.001;%Step of the gradient procedures
for calculation of the Conjugate variables
adx = 0.001; ads = 0.001; adp = 0.001; adv = 0.001;%Steps of the gradient procedures
for calculation of the Interconnections in time

eps1=0.001;
eps2=0.001;%tolerances for stpo of the calculation of the first and second levels
%epsx = 0.01; epss = 0.01; epsp = 0.01; epsv = 0.01;%Conjugate error variables
%epsF = 0.01; epsT = 0.01; epspH = 0.01; epsDO2 = 0.01;%Control error variables

%Initial values of the penalty coefficients
mx=0.001;
ms=0.001;
mp=0.001;
mv=0.001;
astate=[ax as ap av];
acon=[aT aF apH aDO2];
al=[alx als alp alv];
mu=[mx ms mp mv];
%Initial trajectory values of the conjugate and control variables

for k = 1:K
 %Initial trajectories for the conjugate variables
        lx(k)=0.001;
        ls(k)=0.005;
        lp(k)=0.002;
        lv(k)=0.003;
end
l=[lx;ls;lp;lv];

%Initial trajectories of the control variables
T=16;
pH=4.8;
DO2=37;
F=0.;


% Model co-efficients
a1 = 0.0004; a2 = 0.0001;a3 = 0.0002; a4 = 0.0001; a5 = 0.0001; a6 = 0.00002; a7 =
0.00002;
%b1 = 0.2; b2 = 0.03;b3 = 0.01; b4 = 0.01; b5 = 0.001; b6 = 0.002; b7 = 0.001;
```

```
b1 = 4; b2 = 0.5 ;b3 = 0.13; b4 = 0.13; b5 = 0.013; b6 = 0.014; b7 = 0.013;
%c1 = 2; c2 = 0.3; c3 = 0.021; c4 = 0.051; c5 = 0.00211; c6 = 0.0211; c7 = 0.002;
c1 = 0.0002; c2 = 0.0001; c3 = 0.000021; c4 = 0.00021; c5 = 0.000211; c6 =
0.0000211; c7 = 0.0002;
%d1 = 0.0040; d2 = 0.006; d3 = 0.001; d4 = 0.005; d5 = 0.0005; d6 = 0.001; d7 =
0.0002;
d1 = 0.0360; d2 = 0.008; d3 = 0.003; d4 = 0.005; d5 = 0.005; d6 = 0.001; d7 =
0.0002;
par = [a1 a2 a3 a4 a5 a6 a7 b1 b2 b3 b4 b5 b6 b7 c1 c2 c3 c4 c5 c6 c7 d1 d2 d3 d4 d5
d6 d7];

kd = 0.00008;
pl = 50;
m  = 0.11;
n  = 0.52;
si= 266.6;
dt = 2.5;%Delta t

%Initial values of state variables
x0=1.83;
s0=150.0;
p0=2.1;
v0=2.1;
y0=[x0 s0 p0 v0];

%Constraints of state and control variables
Tmin = 12;   %%%%%%%%%%%%%%%%%%%%%%%%%
pHmin = 4;  %                         %
DO2min = 10; %                         %
Fmin = 0;   % For control variables%
Tmax = 20; %                         %
pHmax = 7; %                         %
DO2max = 60;%                         %
Fmax = 50;  %%%%%%%%%%%%%%%%%%%%%%%%%
% State variables%
vmin = 2;
vmax = 5;
pmin = 2.0;
pmax = 70;
smin = 20;
smax = 300;
xmin = 0.83;
xmax =55;

conmin=[Tmin pHmin DO2min Fmin];
conmax=[Tmax pHmax DO2max Fmax];
statemin=[xmin smin pmin vmin];
statemax=[xmax smax pmax vmax];

%Calculation of the initial trajectories of state variables
t=1:K+1
[t,y]=ode15s('model_func1',t,y0)
x=(y(:,1))';
s=(y(:,2))';
```

```
p=(y(:,3))';
v=(y(:,4))'; %transformation to raw vectors

figure(1)
plot(t,x,'-k',t,s,':k',t,p,'-.k',t,v,'--k'),
title('Initial trajectories of state variables');
ylabel('g/l');
xlabel('time in samples');
legend('biomass','substrate','product','volume','Location','NorthEast')



for k = 1:K
   % Calculation of the initial trajectories of the interconnections in time domain
    dx(k) = 10.;
    ds(k) = 138.0;
    dp(k) = 25.0;
    dv(k) = 2.0;
end

T=16.0*ones(1,K);
pH=4.8*ones(1,K);
DO2=37*ones(1,K);
F=5.0*ones(1,K);

 %Calculation of the initial trajectories of Error_Lambda_v_(k)
 for k=1:K
    elx(k) = dx(k)-x(k+1);
    els(k) = ds(k)-s(k+1);
    elp(k) = dp(k)-p(k+1);
    elv(k) = dv(k)-v(k+1);
 end
 el=[elx;els;elp;elv];



%The function subprog.m is on the client path, but it has to be availlable
%to the workers.One way to do this is to use the FileDepandencies property
%of the pjob.

% Start of the coordinating procedure
j=1
while j <= M2;%Second level iterations
jm=findResource('scheduler','type','jobmanager',...
    'name','Myjobmanager1','LookupURL','TzonevaRa')
pjob=createParallelJob(jm)
set(pjob,'MinimumNumberOfWorkers',8)
set(pjob,'MaximumNumberOfWorkers',8)
set(pjob,'FileDependencies',{'subprob2.m'})

%Starting parallel computing

%Create task- only one with 8 output arguments  and not input arguments
task=createTask(pjob, @subprob2,8,{})
```

```
submit(pjob)

%wait for the end of calculation
waitForState(pjob,'finished')
results=getAllOutputArguments(pjob)
get(pjob,'StartTime')
get(pjob,'FinishTime')
%Calculation of the improved values of the coordinating variables
%Calculation of the errors e_lambda_V (k)

for k=1:K
    elx(k) = dx(k) - x(k+1);%Equation 5.25
    els(k) = ds(k) - s(k+1);%Equation 5.26
    elp(k) = dp(k) - p(k+1);%Equation 5.27
    elv(k) = dv(k) - v(k+1);%Equation 5.28
end
el=[elx;els;elp;elv];
%Calculation of the norm of the errors
nn1=norm(elx)+ norm(els)+ norm(elp)+ norm(elv);
%Check of the convergency of the coordinating procedure for the end of the
%calculations
if nn1<=eps2
    break

else

for k =1:K
    lx(k) = lx(k) - alx*elx(k);
    ls(k) = ls(k) - als*els(k);
    lp(k) = lp(k) - alp*elp(k);
    lv(k) = lv(k) - alv*elv(k);

    dx(k) = dx(k) + adx*mx*elx(k);
    ds(k) = ds(k) + ads*ms*els(k);
    dp(k) = dp(k) + adp*mp*elp(k);
    dv(k) = dv(k) + adv*mv*elv(k);
end
end
l=[lx;ls;lp;lv];
%destroy(pjob)
j=j+1
end
%Plot Results
figure(2)
t=1:K+1
plot(t,x,'-k',t,p,':k',t,v,'-.k')
title('Optimised Results for the biomass, product and volume states')
ylabel('g/l');
xlabel('time in samples');
legend('biomass','product','volume','Location','NorthWest')

figure(3)
title('Optimised Results for the substrate')
ylabel('g/l');
```

```
xlabel('time in samples');
plot(s,'-k')
legend('substrate','Location','NorthEast')
t1=1:K
figure(4)
plot(t1,T,'-k',t1,pH,':k',t1,DO2,'-.k',t1,F,'--k')
title('Optimised Results for the Temperature T, pH, DO2,input flow rate F')
ylabel('g/l');
xlabel('time in samples');
legend('Temperature T', 'pH', 'DO2','Input flow rate
F','Location','NorthEastOutside')

figure(5)
plot(t1,lx,'-k',t1,ls,':k',t1,lp,'-.k',t1,lv,'--k')
title('Optimised Results for the conjugate variables lambda')
ylabel('g/l');
xlabel('time in samples');
legend('Lambda x','Lambda s','Lambda p','Lambda v','Location','SouthWest')

%Experimental results from the Department of Chemical Enginering without
optimization
xe =
[0.83,0.85,0.86,0.88,0.9,0.91,0.92,0.92,0.93,1,1.13,1.2,1.33,1.44,1.55,1.66,1.77,2.2
2,2.65,3.21,3.53,4.09,4.52,5.02,5.5,6.13,6.92,7.74,8.33,9.05,9.75,10.45,11.17,11.86,
12.55,13.22,13.93,14.02,14.02,14.05,14.1,14.16,14.2,14.3,14.3,14.32,14.35,14.39,14.4
];
se =
[154,153.3,152.27,151.9,151.33,150.7,150,149.3,148.67,147.7,146.83,145.6,145,143.8,1
42.67,141.5,140.33,138.3,136,133.8,131.67,129.1,127,124.7,122.33,120.2,118.17,116.1,
114,112.2,109.33,106.9,104.67,101.5,98.5,95.4,92.33,92.8,93.42,94,94.5,94.4,94.35,94
.3,94.2,94.3,94.4,94.5,94.6];
pe =
[2.1,2.18,2.28,2.36,2.45,2.53,2.63,2.71,2.8,3.15,3.5,3.85,4.2,4.55,4.9,5.25,5.6,6.78
,7.99,9.16,10.38,11.51,12.76,13.97,15.15,16.67,18.21,19.98,21.28,22.8,24.34,25.84,27
.4,28.73,30.05,30,29.03,28.74,28.52,28.26,28,27.7,27.5,27.3,27,26.8,26.5,26.3,26]
k = 1:K+1;
figure(6)
plot (k,xe,'-k',k,x,'--k')
legend('Experiment biomass','Optimal biomass','Location','NorthWest')
title('Comparison between experimental and Optimal trajectories')
ylabel('g/l');
xlabel('time in samples');
hold off
figure(7)
plot (k,se,'-k',k,s,'--k')
title('Comparison between experimental and Optimal trajectories')
legend('Experiment substrate','Optimal substrate','Location','NorthEast')
ylabel('g/l');
xlabel('time in samples');
hold off
figure(8)
plot (k,pe,'-k',k,p,'--k')
ylabel('g/l');
xlabel('time in samples');
```

```
title('Comparison between experimental and Optimal trajectories')
legend('Experiment product','Optimal product','Location','NorthWest')
hold off
figure(9)
plot(k,xe,'-k',k,se,'--k',k,pe,'-.k')
ylabel('g/l');
xlabel('time in samples');
title('Experimental tragectories')
legend('Biomass','Substrate','Product', 'Location','NorthEast')
```

```
function [x,s,p,v,T,pH,DO2,F]=subprob2
global par T pH DO2 F kd m pl n si x1 s1 p1 v1 T1 pH1 DO21 F1 K M1 eps1 mu l x s p v el conmin conmax statemin statemax astate acon dt y0

%Calculation of the initial and final point of the subintervals
dd=K/numlabs
a=(labindex-1)*dd+1
b=labindex*dd

for k=a:b %Decomposition in time domain
    i=1
    while i <= M1 %First level iterations

    dmmaxdT(k) = par(2)+2*par(3)*T(k);
    dKsdT(k)=par(9)+2*par(10)*T(k);%Equation 5.33
    dYxsdT(k)=par(16)+2*par(17)*T(k);
    dYpxdT(k)=par(23)+2*par(24)*T(k);  % for the temperature

    dmmaxdpH(k)=par(4)+2*par(5)*pH(k);
    dKsdpH(k)=par(11)+2*par(12)*pH(k);
    dYxsdpH(k)=par(18)+2*par(19)*pH(k);
    dYpxdpH(k)=par(25)+2*par(26)*pH(k);  % for the pH

    dmmaxdDO2(k) =par(6)+2*par(7)*DO2(k);
    dKsdDO2(k) =par(13)+2*par(14)*DO2(k);
    dYxsdDO2(k) = par(20)+2*par(21)*DO2(k);
    dYpxdDO2(k) = par(27)+2*par(28)*DO2(k);  % for the DO2

    % Calculation of the kinetic coefficients

    mmax(k)=par(1)+par(2)*T(k)+par(3)*(T(k)^2)+par(4)*pH(k)+par(5)*(pH(k)^2)+par(6)*DO2(k)+par(7)*(DO2(k)^2);
    Ks(k)=par(8)+par(9)*T(k)+par(10)*(T(k)^2)+par(11)*pH(k)+par(12)*(pH(k)^2)+par(13)*DO2(k)+par(14)*(DO2(k)^2);
    Yxs(k)=par(15)+par(16)*T(k)+par(17)*(T(k)^2)+par(18)*pH(k)+par(19)*(pH(k)^2)+par(20)*DO2(k)+par(21)*(DO2(k)^2);
    Ypx(k)=par(22)+par(23)*T(k)+par(24)*(T(k)^2)+par(25)*pH(k)+par(26)*(pH(k)^2)+par(27)*DO2(k)+par(28)*(DO2(k)^2);

% Calculation of the expressions lambda_v_(k), k=0,K-1, v = x,s,p,v and lamda1(k), lamda2(k)


ff(k)=dt*mmax(k) *((1-p(k)/pl)^0.52);%Equation 5.48
fx(k)=ff(k)*s(k)/(Ks(k) + s(k));
fs(k)=ff(k)*Ks(k) * x(k)/((Ks(k) + s(k))^2);%Equation 5.49
fp(k)=dt*mmax(k) * Ypx(k) *((1-p(k)/pl)^-0.48)*x(k)*s(k)/(pl*(Ks(k) + s(k)));% Equation 5.51
fv(k)=dt*F(k)/(v(k)^2);%Equation 5.50

f1(k)=ff(k)*x(k)*s(k)/(Ks(k) + s(k));%Equation 5.52
f2(k)=f1(k)/(Ks(k) + s(k));%Equation 5.5

% Calculation of the expressions ee_v_(k)=lambda(k)-m_v*E_lambda_v_(k)
```

```
eex(k)=l(1,k)-mu(1)*el(1,k);
ees(k)=l(2,k)-mu(2)*el(2,k);
eep(k)=l(3,k)-mu(3)*el(3,k);
eev(k)=l(4,k)-mu(4)*el(4,k);

%Calculation of the gradients for state variables
ex(k)=(1+fx(k)-dt*kd+dt*F(k)/v(k))*eex(k)+((fx(k)/Yxs(k))- dt*m)*ees(k)+dt*fx(k)*Ypx
(k) * eep(k);%Equation 5.54
es(k)=fs(k)*eex(k)+(1-fs(k)/Yxs(k) - dt*F(k)/v(k))*ees(k)+Ypx(k) * fs(k)*eep(k);%
Equation 5.55
ep(k)=(1+fp(k)-dt*F(k)/v(k))*eep(k);%Equation 5.56
ev(k)=fv(k)*x(k)*eex(k)-(fv(k)*(si-s(k))*ees(k))+(fv(k)*p(k)*eep(k))+ eev(k);%
Equation5.57

% Calculation of the gradients for control variables
eF(k) = dt*x(k)/v(k)*eex(k)+(dt*(si-s(k))/v(k)*ees(k))-(dt*p(k)/v(k)*eep(k))+dt*eev
(k);%Equation 5.58
eT(k) = (f1(k)*dmmaxdT(k)-f2(k)*dKsdT(k))*eex(k) + (-f1(k))*(dmmaxdT(k)*Yxs(k)-mmax
(k)*dYxsdT(k)/(Yxs(k)^2))+ ((f2(k)/Yxs(k))*dKsdT(k)*ees(k)) + (f1(k)*(dYpxdT(k)*mmax
(k)+Ypx(k)*dmmaxdT(k)))-(Ypx(k)*f2(k)*dKsdT(k))*eep(k);%Equation 5.59
epH(k) = (f1(k)*dmmaxdpH(k)-f2(k)*dKsdpH(k))*eex(k) + (-f1(k))*(dmmaxdpH(k)*Yxs(k)-
mmax(k)*dYxsdpH(k)/(Yxs(k)^2))+((f2(k)/Yxs(k))*dKsdpH(k)*ees(k))+(f1(k)*(dYpxdpH(k)
*mmax(k)+Ypx(k)*dmmaxdpH(k))-(Ypx(k)*f2(k)*dKsdpH(k))*eep(k));%Equation 5.60
eDO2(k) = (f1(k)*dmmaxdDO2(k)-f2(k)*dKsdDO2(k))*eex(k) + (-f1(k))*(dmmaxdDO2(k)*Yxs
(k)-mmax(k)*dYxsdDO2(k)/(Yxs(k)^2))+((f2(k)/Yxs(k))*dKsdT(k)*ees(k))+(f1(k)*
(dYpxdDO2(k)*mmax(k)+Ypx(k)*dmmaxdDO2(k))-(Ypx(k)*f2(k)*dKsdDO2(k))*eep(k));%
Equation 5.61
n1=[ex(k) es(k) ep(k) ev(k) eF(k) eT(k) epH(k) eDO2(k)]';

%Calculation norm of gradients

nn = norm(n1)
%Check the condition for end of the iterations on the first level
if nn <=eps1
break
%Calculation of the new values of the state and control variables
elseif k == 1
        x1(1,k,labindex)=y0(1);
        s1(1,k,labindex)=y0(2);
        p1(1,k,labindex)=y0(3);
        v1(1,k,labindex)=y0(4);
        T1(1,k,labindex) = T(k) + acon(1)*eT(k)
        pH1(1,k,labindex) = pH(k) + acon(2)*epH(k);
        DO21(1,k,labindex) = DO2(k) + acon(3)*eDO2(k);
        F1(1,k,labindex) = F(k) + acon(4)*eF(k);

else
    x1(1,k,labindex) = x(k) + astate(1)*ex(k);
    s1(1,k,labindex) = s(k) + astate(2)*es(k);
    p1(1,k,labindex) = p(k) + astate(3)*ep(k);
    v1(1,k,labindex) = v(k) + astate(4)*ev(k);
    T1(1,k,labindex) = T(k) + acon(1)*eT(k)
    pH1(1,k,labindex) = pH(k) + acon(2)*epH(k);
```

```
    DO21(1,k,labindex) = DO2(k) + acon(3)*eDO2(k);
    F1(1,k,labindex) = F(k) + acon(4)*eF(k);
    end
i=i+1
end

%Check against constraints

if x1(1,k,labindex) < statemin(1)
    x1(1,k,labindex) = statemin(1);
elseif x1(1,k,labindex) > statemax(1)
    x1(1,k,labindex) = statemax(1);
end
if s1(1,k,labindex) <statemin(2)
    s1(1,k,labindex) = statemin(2);
elseif s1(1,k,labindex) > statemax(2)
    s1(1,k,labindex) = statemax(2);
end
if p1(1,k,labindex) < statemin(3)
    p1(1,k,labindex) = statemin(3);
elseif p1(1,k,labindex) > statemax(3)
    p1(1,k,labindex) = statemax(3);
end
if v1(1,k,labindex) < statemin(4)
    v1(1,k,labindex) = statemin(4);
elseif v1(1,k,labindex) > statemax(4)
    v1(1,k,labindex) = statemax(4);
end

if T1(1,k,labindex) < conmin(1)
    T1(1,k,labindex) = conmin(1);
elseif T1(1,k,labindex) > conmax(1)
   T1(1,k,labindex) = conmax(1);
end
if pH1(1,k,labindex) < conmin(2)
    pH1(1,k,labindex) = conmin(2);
elseif pH1(1,k,labindex) > conmax(2)
    pH1(1,k,labindex) = conmax(2);
end
if DO21(1,k,labindex) < conmin(3)
   DO21(1,k,labindex) = conmin(3);
elseif DO21(1,k,labindex) > conmax(3)
    DO21(1,k,labindex) = conmax(3);
end
if F1(1,k,labindex) < conmin(4)
    F1(1,k,labindex) = conmin(4);
elseif F1(1,k,labindex) > conmax(4)
    F1(1,k,labindex) = conmax(4);
end


end  %end of the iterations in the subproblem time interval (a,b)

labBarrier
```

```
if labindex==1

    x2=reshape(x1,1,prod(size(x1)));
    s2=reshape(s1,1,prod(size(s1)));
    p2=reshape(p1,1,prod(size(p1)));
    v2=reshape(v1,1,prod(size(v1)));
    T2=reshape(T1,1,prod(size(T1)))
    pH2=reshape(pH1,1,prod(size(pH1)));
    DO22=reshape(DO21,1,prod(size(DO21)));
    F2=reshape(F1,1,prod(size(F1)));
    x=x2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):(3*K+b+3*dd)↙
(4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)↙
(7*K+a+7*dd):(7*K+b+7*dd)])
    s=s2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):(3*K+b+3*dd)↙
(4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)↙
(7*K+a+7*dd):(7*K+b+7*dd)])
    p=p2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):(3*K+b+3*dd)↙
(4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)↙
(7*K+a+7*dd):(7*K+b+7*dd)])
    v=v2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):(3*K+b+3*dd)↙
(4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)↙
(7*K+a+7*dd):(7*K+b+7*dd)])
    T=T2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):(3*K+b+3*dd)↙
(4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)↙
(7*K+a+7*dd):(7*K+b+7*dd)])
    pH=pH2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):↙
(3*K+b+3*dd) (4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):↙
(6*K+b+6*dd) (7*K+a+7*dd):(7*K+b+7*dd)])
    DO2=DO22([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):↙
(3*K+b+3*dd) (4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):↙
(6*K+b+6*dd) (7*K+a+7*dd):(7*K+b+7*dd)])
    F=F2([a:b (K+a+dd):(K+b+dd) (2*K+a+2*dd):(2*K+b+2*dd) (3*K+a+3*dd):(3*K+b+3*dd)↙
(4*K+a+4*dd):(4*K+b+4*dd) (5*K+a+5*dd):(5*K+b+5*dd) (6*K+a+6*dd):(6*K+b+6*dd)↙
(7*K+a+7*dd):(7*K+b+7*dd)])

        %Calculation of the final value of x(k)=x(K+1)=xK1;sK1,pK1,vK1
    mmax(K)=par(1)+par(2)*T(K)+par(3)*(T(K)^2)+par(4)*pH(K)+par(5)*(pH(K)^2)+par(6)↙
*DO2(K)+par(7)*(DO2(K)^2)
    Ks(K)=par(8)+par(9)*T(K)+par(10)*(T(K)^2)+par(11)*pH(K)+par(12)*(pH(K)^2)+par↙
(13)*DO2(K)+par(14)*(DO2(K)^2)
    Yxs(K)=par(15)+par(16)*T(K)+par(17)*(T(K)^2)+par(18)*pH(K)+par(19)*(pH(K)^2)+par↙
(20)*DO2(K)+par(21)*(DO2(K)^2)
    Ypx(K)=par(22)+par(23)*T(K)+par(24)*(T(K)^2)+par(25)*pH(K)+par(26)*(pH(K)^2)+par↙
(27)*DO2(K)+par(28)*(DO2(K)^2)
    ff(K)=dt*mmax(K) *((1-p(K)/pl)^0.52)
    f1(K)=ff(K)*x(K)*s(K)/(Ks(K) + s(K))
    xK1=x(K)+f1(K)-dt*kd*x(K)-dt*F(K)*x(K)/v(K);
    sK1=s(K)-f1(K)/Yxs(K)-dt*m*x(K)+dt*F(K)*(si-s(K))/v(K);
    pK1=p(K)+f1(K)*Ypx(K)-dt*F(K)*p(K)/v(K);
    vK1=v(K)+dt*F(K);
    if xK1 <= statemin(1)
        xK1 = statemin(1);
        elseif xK1 >=statemax(1)
```

```
    xK1 = statemax(1);
    end
x(49)=xK1

if sK1 <= statemin(2)
    sK1 = statemin(2);
elseif sK1 >= statemax(2)
    sK1 = statemax(2);
end
s(49)=sK1

if pK1 <= statemin(3)
    pK1 = statemin(3);
elseif pK1 >= statemax(3)
    pK1=statemax(3)
end
 p(49)=pK1

if vK1 <= statemin(4)
    vK1 = statemin(4);
elseif vK1 >= statemax(4)
    vK1 = statemax(4);
end
v(49)=vK1
end
```

Appendix D

Scientific paper

(9 pages)

Optimal control of the fed-batch fermentation process for production of yeast

H. Chen R. Tzoneva

Department of Electrical Engineering, Cape Peninsula University of Technology, Cape Town, South Africa

Abstract: A problem for optimal control of a fed-batch fermentation process for production of yeast is formulated and solved. The problem is based on a criterion for maximum production of biomass at the end of fermentation and on an unstructured mass balance biological model incorporating in the kinetic coefficients the physiochemical variables considered as control inputs. A decomposition method for solution of the problem in two level calculation structure is developed on the basis of an augmented functional of Lagrange. A Matlab programme for calculation is written for the fed-batch process for production of yeast. The programme is used as a part of LabVIEW system for control of the fermentation process.

Keywords: Optimal control, augmented functional of Lagrange, decomposition, fed-batch fermentation, Matlab program

1. INTRODUCTION

Fermentation is the intrinsic capability of the microorganism to perform complex chemical transformations upon compounds by means of the metabolic activity and the action of enzymes. These processes are performed in tanks, called bioreactors, or fermentors and are used in the production of foods, drinks, medicines, chemical products, etc. The fermentation processes have been investigated for the past decades [1]. Although the fermentation has its own quality control systems, sudden unexplained changes occur from time to time in the characteristics of the process, which includes: fermentation time, attenuation, contamination, pH, Temperature, Dissolved Oxygen, volumetric flow rate profiles with time [2]. The fermentation processes are described by two groups of variables--biological: concentrations of substrate, biomass and product; and physiochemical: temperature, pH, DO_2, agitation [2].

The problem for control of fed-batch fermentation process is very important these days because of population increasing and industrial developments. Existing control system for the fermentation processes in the scientific laboratories and industry are based only on the principles of classic control. [3], [4]. Only PID or on/off control is used. The process behavior is not optimized according to existing environmental conditions and the acting disturbances. The fermentation processes have proved to have positive effects in many fields of domestic life and industry, but they still need deeper understanding and improvement. The reason is that the fermentation processes have not been studied fully yet as an object of control. It is not clear:

- Which kind of models are convenient, for optimal control calculation
- Which are dependencies between input and outputs,
- Which process variables are the most significant and in which way they affect the quality of the process or its products,
- What the optimal combination of settings for these significant variables is, according to the goals posed on the process quality and the product quantity.
- There is a lack of methods for measuring of important variables, for modelling and parameter estimation and for optimisation and control calculation.

The study is overcoming the constraints highlighted above in the following way:

1) Unstructured model describing the behaviour of the biological variables identifying the dependencies between physiochemical and biological variables is introduced,
   - → Biological variables of the fermentation process : biomass concentration [x], substrate concentration [s], and product concentration [p] are described on the basis of the systematic approach using mass balance equations.
   - → The influence of physiochemical variables over the behaviour of the biological variables is expressed by representation of the kinetic parameters as a quadratic function of these variables.
2) The considered process variables are:
   - → Inputs used as control signals for the physiochemical variables
     - Base/acid flow rate
     - On/off heating energy
     - Flow rate of oxygen

   Input used as control signals for the

biological variables:

- Flow rate of substrate
- Temperature T
- pH
- DO_2 concentration of the medium in the fermentor

→ Outputs and state space variables

- Concentration of biomass
- Concentration of substrate
- Concentration of product
- Volume of the reactor.

3) The optimal settings of the process variables are obtained by solution of a problem for optimal control, together with the problem for local control in a scheme of repetitive optimisation based on real time model parameter estimation.
4) Measurement and control only of the physiochemical variables is used for the adaptive, optimal control of the process.
5) New method for optimal control calculation is developed.

The paper describes formulation and solution of the problem for optimal control on the basis of the variation approach using a Lagrange's augmented functional. The paper is structured as follows: the fermentation process for production of yeast is described in part 2, formulation of the problem for optimal control is described in part 3, decomposition method for solution of the problem for optimal control is described in part 4, decomposition of the problem for optimal control is described in part 5, solution of the optimal control problem in a two level computing structure is described in part 6, Matlab calculation of the problem for optimal control of the yeast fed-batch fermentation is described in part 7.

2. DESCRIPTION OF THE FED-BATCH FERMENTATION PROCESS FOR PRODUCTION OF YEAST

The fermentation process takes place in tanks called bioreactors or fermentors. Fermentation processes are complex, dynamic autocatalytic processes in which transformation, caused by biological catalysts called enzymes, takes place. Enzymes decrease the activation energy {E} of the reaction catalysed by binding the substrate and forming an enzyme-substrate {ES} complex. Molecular aspects of enzymes-substrate interaction are not yet fully understood, but yet the enzymes require optimal conditions for pH, temperature, ionic strength, etc for their maximum activity [2]. The conclusion from here is that the optimal conditions for these variables have to be found for every fermentation runs. This leads to the necessity of solution of a problem for optimal control in real time.

Yeast is widely used in brewing, baking, wine industry, research etc. The paper considers the growth of yeast cells *saccharomyces cerevisiae* using a fed-batch fermentation process. The study is looking at the optimal operation of the fermentation of yeast used to inoculate fermentation of beer. The aim is to maximise the production of yeast at the end of the process in the presence of oxygen (Aerobic culture).

3. Formulation of the problem for optimal control

The aim in producing yeast, needed, as inoculant in beer fermentation is maximum quantity of biomass, which means maximum concentration of biomass at the end of the fermentation process. The problem is how to influence the process in order to achieve this aim. The possibilities for influence are the input substrate flow rate, temperature, pH and DO_2 concentration, which in this study are considered as control variables. Their optimal trajectories can be obtained through the formulation and solution of a problem for optimal control of the process.

The problem can be formulated in a discrete time domain in such a way: Find the trajectories of the feed flow rate, temperature, pH and dissolved oxygen concentration: $F(k), T(k), pH(k), DO_2(k), k = \overline{0, K-1}$ in such a way that the concentration of biomass at the end of the process

$$J = x(K) \rightarrow \max \tag{1}$$

is maximized under the model equations:

$$x(k+1) = x(k) + \Delta t \mu_{\max}\left[1 - \frac{p(k)}{p^*}\right]^{\alpha}\left[\frac{x(k)s(k)}{K_s + s(k)}\right] - \Delta t k_d x(k) - \Delta t \frac{F(k)x(k)}{V(k)} = f_x(k), x(0) = x_0 \tag{2}$$

$$s(k+1) = s(k) - \frac{\Delta t \mu_{\max}}{Y_{x/s}}\left[1 - \frac{p(k)}{p^*}\right]^{\alpha}\left[\frac{x(k)s(k)}{K_s + s(k)}\right] - \Delta t m x(k) + \Delta t \frac{F(k)}{V(k)}[s_i - s(k)] = f_s(k), s(0) = s_0 \tag{3}$$

$$p(k+1) = p(k) + \Delta t \mu_{\max} Y_{p/x}\left[1 - \frac{p(k)}{p^*}\right]^{\alpha} \cdot \left[\frac{x(k)s(k)}{K_s + s(k)}\right] - \Delta t \frac{F(k)p(k)}{V(k)} = f_p(k), p(0) = 0 \tag{4}$$

$$V(k+1) = V(k) + \Delta t F(k) = f_v(k), \quad V(0) = V_0, \quad V(K) = V_{\max} \tag{5}$$

$$\mu_{\max}(k) = a_1 + a_2 T(k) + a_3 T^2(k) + a_4 pH(k) + a_5 pH^2(k) + a_6 DO_2(k) + a_7 DO^2{}_2(k) \tag{6}$$

$$K_s(k) = b_1 + b_2 T(k) + b_3 T^2(k) + b_4 pH(k) + b_5 pH^2(k) + b_6 DO_2(k) + b_7 DO^2{}_2(k) \tag{7}$$

$$Y_{x/s}(k) = c_1 + c_2 T(k) + c_3 T^2(k) + c_4 pH(k) + c_5 pH^2(k) + c_6 DO_2(k) + c_7 DO_2^2(k) \tag{8}$$

$$Y_{p/x}(k) = d_1 + d_2 T(k) + d_3 T^2(k) + d_4 pH(k) + d_5 pH^2(k) + + d_6 DO_2(k) + d_7 DO_2^2(k) \quad (9)$$

and under limitations over the minimal and maximal values of the process variables

$$\begin{aligned} &x_{min}(k) \le x(k) \le x_{max}(k) \\ &s_{min}(k) \le s(k) \le s_{max}(k) \\ &p_{min}(k) \le p(k) \le p_{max}(k) \\ &V_{min}(k) \le V(k) \le V_{max}(k) \\ &T_{min}(k) \le T(k) \le T_{max}(k) \\ &pH_{min}(k) \le pH(k) \le pH_{max}(k) \\ &DO_{min}(k) \le DO(k) \le DO_{max}(k) \\ &F_{min}(k) \le F(k) \le F_{max}(k) \end{aligned} \quad (10)$$

where k is the discrete time, and K is the final moment of the process,

F	Feed rate for nutrient substrate in $[l/s]$,
x_i	Biomass concentration input flow in $[g/l]$,
s_i	Substrate concentration input in $[g/l]$,
p^*	product concentration in limitation for the inhibition $[g/l]$,
V	Culture volume in the reactor vessel $[l]$,
x	Biomass concentration $[g/l]$,
s	Substrate concentration $[g/l]$,
p	Product concentration, in the fermentor $[g/l]$
μ_{max}	Maximum specific growth rate
K_s	Monod constant or simply saturation constant
K_d	Specific death rate $[h]$
m_s	Maintenance co-efficient
$Y_{x/s}$	Production co-efficient biomass from substrate
$Y_{p/x}$	Production co-efficient product from substrate
s_o	Initial reactor substrate concentration $[g/l]$
x_o	Initial biomass concentration $[g/l]$
p_o	Initial product concentration $[g/l]$
V_o	Initial culture volume in the reactor vessel $[l]$
Δt	Sampling period for the discrete model $[h]$
$\overline{a_1, a_7}, \overline{b_1, b_7}, \overline{c_1, c_7}, \overline{d_1, d_7}$	Model coefficients
n	Power coefficient

The problem (1)-(10) is solved using the variational approach and decomposition.

4. DECOMPOSITION METHOD FOR SOLUTION OF THE PROBLEM FOR OPTIMAL CONTROL

The problem (1)-(10) is a non-convex one with nonlinear model and constraints over the minimum and maximum values of the variables. This leads to many difficulties in solving the optimal control problems as a non-linear two point boundary value problem has to be solved and singular kind of control to be calculated [5]. A decomposition method for solving the discrete time formulation of the problem is proposed in the paper based on Variation calculus and using an augmented Lagrange's functional and its decomposition in time domain. The method is characterized by a new coordinating vector that permits an augmented Lagrange's functional to be used for solving the non-convex optimal control problem, and for preserving the full decomposition in time domain of the dual problem. The optimal profiles of the physiochemical variables are the end solution of the optimization problem. They present the desired outputs for the local PID control loops for the temperature, pH, dissolved oxygen, etc. These trajectories can be sent as set-points for the controllers of T, pH, DO, F.

4.1 Augmented functional of Lagrange

The used augmented functional of Lagrange has the form:

$$\begin{aligned} L = x(k) + \sum_{k=0}^{K-1} \{ &\lambda_x(k)[-x(k+1) + f_x(k)] + \lambda_s(k)[-s(k+1) \\ &+ f_s(k)] + \lambda_p(k)[-p(k+1) + f_p(k)] + \lambda_v(k)[-v(k+1) + \\ &f_p(k)] + \mu_x[-x(k+1) + f_x(k)]^2 + \mu_s[-s(k+1) + f_s(k)]^2 \\ &\mu_p[-p(k+1) + f_p(k)]^2 + \mu_v[-v(k+1) + f_v(k)]^2 \} \end{aligned} \quad (11)$$

where λ are the costate variables and μ are the penalty coefficients.

The functional of Lagrange is augmented with the quadratic terms of the model equations in order to convexify the problem for optimal control calculation. The values of the penalty coefficients μ play a role in the process of calculation influencing the rate of convergence of the calculating procedures.

The augmented functional has to be maximized according to state and control variables and minimized according to the conjugate variables on the basis of the theory of duality, under the process constraints represented by equations (10).

4.2. Necessary conditions for optimality

The solution of the optimal control problem is based on the necessary conditions for the optimality of the functional of Lagrange.

→ For the state variables:

$$\frac{\partial L}{\partial x(k)} = 0, \quad \frac{\partial L}{\partial s(k)} = 0, \quad \frac{\partial L}{\partial p(k)} = 0, \quad \frac{\partial L}{\partial V(k)} = 0, \quad (12)$$

→ For the control variables

$$\frac{\partial L}{\partial T(k)} = 0, \quad \frac{\partial L}{\partial pH(k)} = 0, \quad \frac{\partial L}{\partial DO_2(k)} = 0, \quad \frac{\partial L}{\partial F(k)} = 0, \quad (13)$$

→ For the conjugate variables

$$\frac{\partial L}{\partial \lambda_x(k)} = 0, \quad \frac{\partial L}{\partial \lambda_s(k)} = 0, \quad \frac{\partial L}{\partial \lambda_p(k)} = 0, \quad \frac{\partial L}{\partial \lambda_v(k)} = 0. \quad (14)$$

The analytical expressions for the derivatives are represented in the following way:

→ For the state variables

$$\frac{\partial L}{\partial x(k)} = e_x(k) = \left[1 + \varphi_x(k) - \Delta t k_d + \Delta t \frac{F(k)}{V(k)}\right] \cdot$$
$$\cdot \left[\lambda_x(k) - \mu_x(k) e_{\lambda x}(k)\right] + \left[\frac{e_x(k)}{Y_{x/s}} - \Delta t m\right] \cdot$$
$$\cdot \left[\lambda_s(k) - \mu_s(k) e_{\lambda s}(k)\right] + \Delta t \varphi_x(k) Y_{p/x} \cdot$$
$$\cdot \left[\lambda_p(k) - \mu_p(k) e_{\lambda p}(k)\right] \tag{15}$$

$$\frac{\partial L}{\partial s(k)} = e_s(k) = \varphi_s(k)\left[\lambda_x(k) - \mu_x(k) e_{\lambda x}(k)\right] +$$
$$+ \left[1 - \frac{\varphi_x(k)}{Y_{x/s}} - \Delta t \frac{F(k)}{V(k)}\right]\left[\lambda_s(k) - \mu_s(k) e_{\lambda s}(k)\right] +$$
$$+ Y_{p/x} \varphi_s(k)\left[\lambda_p(k) - \mu_p(k) e_{\lambda p}(k)\right] \tag{16}$$

$$\frac{\partial L}{\partial p(k)} = e_p(k) = \left[1 + \varphi_p(k) - \Delta t \frac{F(k)}{V(k)}\right] \cdot$$
$$\cdot \left[\lambda_p(k) - \mu_p(k) e_{\lambda p}(k)\right] \tag{17}$$

$$\frac{\partial L}{\partial V(k)} = e_V(k) = \varphi_V(k) x(k)\left[\lambda_x(k) - \mu_x(k) e_{\lambda x}(k)\right] -$$
$$- \varphi_V(k)\left[s_i - s(k)\right]\left[\lambda_s(k) - \mu_s(k) e_{\lambda s}(k)\right] +$$
$$+ \varphi_V(k) p(k)\left[\lambda_p(k) - \mu_p(k) e_{\lambda p}(k)\right] +$$
$$+ \left[\lambda_V(k) - \mu_V(k) e_{\lambda V}(k)\right] \tag{18}$$

→ For the control variables

$$\frac{\partial L}{\partial F(k)} = e_F(k) =$$
$$= \frac{\Delta t x(k)}{V(k)}\left[\lambda_x(k) - \mu_x(k) e_{\lambda x}(k)\right] + \frac{\Delta t[s_i - s(k)]}{V(k)} \cdot$$
$$\cdot \left[\lambda_s(k) - \mu_s(k) e_{\lambda s}(k)\right] - \frac{\Delta t p(k)}{V(k)}\left[\lambda_p(k) - \mu_p(k) e_{\lambda p}(k)\right] +$$
$$+ \Delta t\left[\lambda_V(k) - \mu_V(k) e_{\lambda V}(k)\right] \tag{19}$$

$$\frac{\partial L}{\partial T(k)} = e_T(k) =$$
$$= \left[\varphi_1(k) \frac{\partial \mu_{max}}{\partial T} - \varphi_2(k) \frac{\partial K_s}{\partial T}\right]\left[\lambda_x(k) - \mu_x(k) e_{\lambda x}(k)\right] +$$
$$+ \left[\left[-\varphi_1(k)\right]\left[\frac{\frac{\partial \mu_{max}}{\partial T} Y_{x/s} - \mu_{max} \frac{\partial Y_{x/s}}{\partial T}}{Y_{x/s}^2}\right] +\right.$$
$$\left. + \frac{\partial_2(k)}{Y_{x/s}} \frac{\partial K_s}{\partial T}\right]\left[\lambda_s(k) - \mu_s(k) e_{\lambda s}(k)\right] +$$
$$\left[\varphi_1(k)\left[\frac{\partial Y_{p/x}}{\partial T} \mu_{max} + Y_{p/x} \frac{\partial \mu_{max}}{\partial T}\right] - Y_{p/x} \varphi_2(k) \frac{\partial K_s}{\partial T}\right]$$
$$\left[\lambda_p(k) - \mu_p(k) e_{\lambda p}(k)\right] \tag{20}$$

$$\frac{\partial L}{\partial pH} = e_{pH}(k) =$$
$$= \left[\varphi_1(k) \frac{\partial \mu_{max}}{\partial pH} - \varphi_2(k) \frac{\partial K_s}{\partial pH}\right]\left[\lambda_x(k) - \mu_x(k) e_{\lambda x}(k)\right] +$$
$$+ \left[\left[-\varphi_1(k)\right]\left[\frac{\frac{\partial \mu_{max}}{\partial pH} Y_{x/s} - \mu_{max} \frac{\partial Y_{x/s}}{\partial pH}}{Y_{x/s}^2}\right] + \frac{\partial_2(k)}{Y_{x/s}} \frac{\partial K_s}{\partial pH}\right]$$
$$\left[\lambda_s(k) - \mu_s(k) e_{\lambda s}(k)\right] + \left[\varphi_1(k)\left[\frac{\partial Y_{p/x}}{\partial pH} \mu_{max} + Y_{p/x} \frac{\partial \mu_{max}}{\partial pH}\right]\right.$$
$$\left. - Y_{p/x} \varphi_2(k) \frac{\partial K_s}{\partial pH}\right]\left[\lambda_p(k) - \mu_p(k) e_{\lambda p}(k)\right] \tag{21}$$

$$\frac{\partial L}{\partial DO_2} = e_{DO_2}(k) =$$
$$= \left[\varphi_1(k) \frac{\partial \mu_{max}}{\partial DO_2} - \varphi_2(k) \frac{\partial K_s}{\partial DO_2}\right]\left[\lambda_x(k) - \mu_x(k) e_{\lambda x}(k)\right]$$
$$+ \left[\left[-\varphi_1(k)\right]\left[\frac{\frac{\partial \mu_{max}}{\partial DO_2} Y_{x/s} - \mu_{max} \frac{\partial Y_{x/s}}{\partial DO_2}}{Y_{x/s}^2}\right] + \frac{\partial_2(k)}{Y_{x/s}} \frac{\partial K_s}{\partial T}\right]$$
$$\left[\lambda_s(k) - \mu_s(k) e_{\lambda s}(k)\right] + \left[\varphi_1(k)\left[\frac{\partial Y_{p/x}}{\partial DO_2} \mu_{max} + Y_{p/x} \frac{\partial \mu_{max}}{\partial DO_2}\right]\right.$$
$$\left. - Y_{p/x} \varphi_2(k) \frac{\partial K_s}{\partial DO_2}\right]\left[\lambda_p(k) - \mu_p(k) e_{\lambda p}(k)\right] \tag{22}$$

→ For the conjugate variables

$$\frac{\partial L}{\partial \lambda_x(k)} = x(k+1) - f_x(k) = 0 = e_{\lambda x}(k) \tag{23}$$

$$\frac{\partial L}{\partial \lambda_s(k)} = s(k+1) - f_s(k) = 0 = e_{\lambda s}(k) \tag{24}$$

$$\frac{\partial L}{\partial \lambda_p(k)} = p(k+1) - f_p(k) = 0 = e_{\lambda p}(k) \tag{25}$$

$$\frac{\partial L}{\partial \lambda_v(k)} = V(k+1) - f_v(k) = 0 = e_{\lambda v}(k) \tag{26}$$

where

$$\Delta t \mu_{max}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{s(k)}{K_s + s(k)} = \varphi_x(k) \tag{27}$$

$$\Delta t \mu_{max}\left[1 - \frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{\left[K_s + s(k)\right]^2} = \varphi_s(k) \tag{28}$$

$$\Delta t \frac{F(k)}{V^2(k)} = \varphi_v(k) \tag{29}$$

$$\frac{\Delta t \mu_{max} Y_{p/x}}{p^*}\left[1 - \frac{p(k)}{p^*}\right]^{n-1} \frac{x(k)s(k)}{K_s + s(k)} = \varphi_p(k) \tag{30}$$

$$\Delta t\left[1-\frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{K_s+s(k)}=\varphi_1(k) \tag{31}$$

$$\Delta t\mu_{max}\left[1-\frac{p(k)}{p^*}\right]^n \frac{x(k)s(k)}{[K_s+s(k)]^2}=\varphi_2(k), k=\overline{0,K-1} \tag{32}$$

and the partial derivatives of the kinetic parameters are:

→ According to the temperature

$$\left.\begin{aligned}\frac{\partial\mu_{max}}{\partial T}&=a_2+2a_3T(k),\ \frac{\partial K_s}{\partial T}=b_2+2b_3T(k),\\ \frac{\partial Y_{x/s}}{\partial T}&=c_2+2c_3T(k),\ \frac{\partial Y_{p/s}}{\partial T}=d_2+2d_3T(k),\end{aligned}\right\} \tag{33}$$

→ According to the DO_2.

$$\left.\begin{aligned}\frac{\partial\mu_{max}}{\partial DO_2}&=a_6+2a_7DO_2,\ \frac{\partial K_s}{\partial DO_2}=b_6+2b_7DO_2(k),\\ \frac{\partial Y_{x/s}}{\partial DO_2}&=c_6+2c_7DO_2(k),\ \frac{\partial Y_{p/s}}{\partial DO_2}=d_6+2d_7DO_2(k),\end{aligned}\right\} \tag{34}$$

→ According to the pH.

$$\left.\begin{aligned}\frac{\partial\mu_{max}}{\partial pH}&=a_4+2a_5pH(k),\ \frac{\partial K_s}{\partial pH}=b_4+2b_5pH(k),\\ \frac{\partial Y_{x/s}}{\partial pH}&=c_4+2c_5pH(k),\ \frac{\partial Y_{p/s}}{\partial pH}=d_4+2d_5pH(k),\end{aligned}\right\} \tag{35}$$

$k=\overline{0,K-1}$

5. DECOMPOSITION OF THE PROBLEM FOR OPTIMAL CONTROL

The optimal trajectories of the variables can be calculated as a solution of the obtained set of non-linear, algebraic equations (15)-(35). It can be seen that the number of equations and number of variables is very big. The solution could be found easier if the problem is decomposed. The decomposition in time domain could reduce complexity of the system of equations [6], [7]. The solutions in time domain will consists of *K+1* separate solutions, every one at separate time moment *k*. The decomposition in time domain can be obtained on the basis of the coordinating procedure in two level computing structure using the conjugate variables λ as coordinating ones [6], but this method is applicable for normal functional of Lagrange. The used augmented functional could not be fully decomposed because of the quadratic terms and variables *x(k+1), s(k+1), p(k+1) v(k+1)* in them. They play a role of interconnections in time domain [8]. This type of functional can be fully decomposed if the considered variables are selected also as a part of the coordinating vector.

If it is denoted,

$$\delta_v(k)=v(k+1), v=x,s,p,V, k=\overline{0,K-1}, \tag{36}$$

then the coordinating for time domain decomposition vector will be

$$\lambda_v(k)=\lambda_v^j, \tag{37}$$

$$\delta_v(k)=\delta_v^j, v=x,s,p,v,\ \ k=\overline{0,K-1}, \tag{38}$$

where *j* is an index of iterations in the coordinating procedure. If equations (37)-(38) are substituted in the augmented functional of Lagrange, the functional and the necessary conditions for optimality are fully decomposed in time domain. The common problem is decomposed into one coordinating sub-problem and *K+1* sub-problems on the first level of the calculating two level structure, Fig 1:

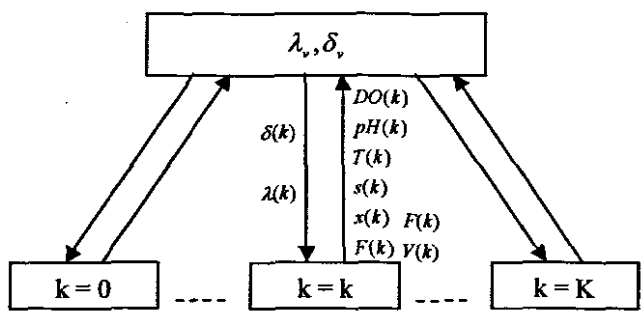


Figure 1: Two level calculating structures

6. SOLUTION OF THE OPTIMAL CONTROL PROBLEM IN A TWO LEVEL COMPUTING STRUCTURE

6.1. First level sub-problems

The solutions of the first level sub-problems represent solution of the set of equations (15)-(22), (27) – (35) in which the coordinating variables are substituted. These equations are solved for every separate moment of time in a parallel way. It can be seen from the structure of the equations that analytical solution is impossible. The gradient procedure can be used in the following way

$$v^{j+1}(k)=v^j(k)+\alpha_v e_v^j(k), v=x,s,p,v,F,T,pH,DO_2 \tag{39}$$

where $\alpha_v>0$ are the steps of the gradient procedures, $e_v(k)$ are the gradients according to the equations (15)-(22) and *i* is the iteration index of the procedures.

The calculation of the gradients is done with the old values of the state and control variables v^j, $v=x,s,p,V,F,T,pH,DO_2$. The calculation with the gradient procedure will continue until the necessary conditions for optimality towards state and control variables are fulfilled. This means that the values of the gradients are approximately equal to zero:

$$\|e_v(k)\|\le\varepsilon_v, \varepsilon_v>0, v=x,s,p,V,F,T,pH,DO_2, k=\overline{0,K-1} \tag{40}$$

The new calculated values have to be checked for belonging to the area of constraints (10). Simple way to do and correct the obtained new values is to project them over the constraints domain according to the relation:

$$v^{j,j}(k)=\begin{Bmatrix} v_{\min},\ if\ v^{j,j}(k)<v_{\min} \\ v^{j,j}(k),\ if\ v_{\min}\le v^{j,j}(k)\le v_{\max},\ k=\overline{0,K-1} \\ v_{\max}\ if\ v^{j,j}(k)>v_{\max} \end{Bmatrix}$$

$where\ v=x,s,p,V,F,T,pH,DO_2$

(41)

The obtained values of the trajectories of the state and control variables are function of the coordinating variables. They are optimal for the given values of the coordinating variables. If the coordinating trajectories are optimal ones, then the obtained state and control variables also will be the optimal according to the theory of the duality in optimal control and Lagrange methods.

6.2. The coordinating sub-problem

The coordinating variables are obtained from the necessary conditions for optimality of the augmented Lagrange functional according to them. For the conjugate variables these equations are (23)-(26). The necessary conditions for optimality according to the new introduced interconnections in time domain are:

$$\frac{\partial L}{\partial \delta_x(k)}=\mu_x\left[\delta_x(k)-f_x(k)\right]=e_{\delta x}(k)=0=\mu_x e_{\lambda x}(k),$$

(42)

$$\frac{\partial L}{\partial \delta_s(k)}=\mu_s\left[\delta_s(k)-f_s(k)\right]=e_{\delta s}(k)=0=\mu_s e_{\lambda s}(k),$$

(43)

$$\frac{\partial L}{\partial \delta_p(k)}=\mu_p\left[\delta_p(k)-f_p(k)\right]=e_{\delta p}(k)=0=\mu_p e_{\lambda p}(k),$$

(44)

$$\frac{\partial L}{\partial \delta_V(k)}=\mu_V\left[\delta_V(k)-f_V(k)\right]=e_{\delta V}(k)=0=\mu_p e_{\lambda V}(k),$$

(45)

$$where\ \delta_v(k)-f_v(k)=e_{\lambda v}(k), v=x,s,p,V\ ,$$

(46)

according to equations (23)-(26).

The necessary conditions for optimality (23)-(26) and (38)-(42) can be combined in a gradient procedure. Equations (38)-(42) give the values of the gradients for the conjugate variables multiplied by the penalty coefficient. $\mu_v,\ v=x,s,p,V$.

It appears that:

$$\frac{\partial L}{\partial \delta_v(k)}=\mu_v\frac{\partial L}{\partial_{\lambda v}(k)}=\mu_v e_{\lambda v}(k),\quad k=\overline{0,K-1} \qquad (47)$$

Gradient procedure can be built for both type of coordinating variables.

→ For conjugate variables:

$$\lambda_v(k)=\lambda_v(k)-\alpha_{\lambda v}e_{\lambda v}(k),$$

(48)

$$e_{\lambda v}(k)=\delta_v(k)-f_v(k), v=x,s,p,V, k=\overline{0,K-1},$$

(49)

$$\delta_v(k)=\delta_v(k)+\alpha_{\delta v}e_{\delta v}(k)=\delta_v(k)+\alpha_{\delta v}\mu_v e_v(k),$$

$$k=\overline{0,K-1}$$

(50)

$v=x,s,p,V$

where $\alpha_{\lambda v}$ and $\alpha_{\delta v}$ are the steps of the procedures for calculation of the conjugate variables and the interconnection in time, respectively.

As the calculations are based on gradient iterative procedure, they will stop when the gradient is going to zero, or practically achieving some conditions for optimality, given by

$$\left\|e_{\lambda v}(k)\right\|\le\varepsilon_{\lambda v}, \varepsilon_{\lambda v}>0, k=\overline{0,K-1}, \qquad (51)$$

where $\varepsilon_{\lambda v}$ is a very small positive number.

For every step of the gradient procedure for the coordinating process the full iteration cycle of the sub problems of the first level is done. The calculations of the coordinating variables are done with the values of state and control variables of the first level sub problems and the calculations of the first level sub problems are done with the values of the calculated on the previous iteration coordinating variables. The iterative process of coordination and the optimal solution of the whole problem for optimal control is obtained when the optimal solution of the coordinating sub problem is reached, this means conditions (51) are fulfilled.

6.3. Algorithm for calculation

The calculations in the two-level structure can be grouped and represented in the following algorithm:

1) On the coordinating level the initial values of the coordinating and gradient procedures, initial trajectories of the state, control and coordinating variables are set as follows:
   a) Maximal number of step of the coordinating process – M2.
      Maximal number of steps for state and control variables gradient procedure – M1.
   b) Steps of the gradient procedures
      i. For coordinating conjugate variables - $\alpha_{\lambda v}, v=x,s,p,V$.
      ii. For coordinating interconnection in time - $\alpha_{\delta v}, v=x,s,p,V$.
      iii. For state variables and control variables $\alpha_v, v=x,s,p,V,F,T,pH,DO_2$
   c) Values for the error and termination of the calculations for coordinating procedures. $\varepsilon_{\lambda v},\varepsilon_{\delta v}, v=x,s,p,V$ and for gradient procedures on first level. $\varepsilon_v, v=x,s,p,V$
   d) Initial values trajectories of the conjugate variables.
      $\lambda^1_v(k), v=x,s,p,V, k=\overline{0,K-1}$
   e) Initial values of the state variables, x(0), s(0), p(0), V(0).
   f) Number of steps in the optimization horizon – K.
   g) Initial trajectories of the control variables
      $F^{1,1}(k), T^{1,1}(k), pH^{1,1}(k), DO_2^{1,1}(k),\ k=\overline{0,K-1}$,

h) The initial trajectories of the control variables are substituted in the model equations (2)-(9) and the initial state trajectories are calculated.

i) The initial trajectories of the interconnections in time domain are calculated from $\delta_v(k) = v(k+1), k = \overline{0, K-1}$

j) Initial trajectories of the coordinating variables and initial state and control trajectories are sent to the first level.

2) On the first level the following calculations are done:

a. Derivatives from equation (33)-(35) are calculated.

b. The kinetic coefficients $\mu_{max}, K_s, Y_{x/s}, Y_{p/x}$ from equations (6)-(9) are calculated.

c. The expressions $\varphi_x(k), \varphi_s(k), \varphi_p(k), \varphi_V(k), \varphi_1(k), \varphi_2(k), k = \overline{0, K-1}$ are calculated from equations (27)-(32).

d. The gradients $e_v^{jj}(k), k = \overline{0, K-1}, v = x, s, p, V, F, T, pH, DO_2$ are calculated from equations (15)-(22).

e. The errors for the end of the gradient procedures are calculated according to equation (40). If the error is less than ε_v, the calculations of the first level stop and the current values of the state and control variables are the optimal ones. If the error conditions are not satisfied the new values of the state and control values are calculated.

f. The new values of state and control variables are calculated from equation (39).

g. The obtained state and control variables are projected over the constraint domain according to equation (41). The obtained trajectories are sent to the second level.

3) On the second level the following calculations are done:

a. The gradients $e_{\lambda_v}(k), k = \overline{0, K-1}, v = x, s, p, V$ are calculated from equations (23)-(26).

b. The conditions for the end of the calculations are checked according to Equation (51). If the conditions are fulfilled the optimal solution of the coordinating and of the whole problem are obtained. If the conditions are not satisfied the new values of the coordinating variables are calculated.

c. New values of the conjugate variables are calculated according to equation (48)-(49).

d. The new values of the interconnections are calculated from equation (50). These trajectories are sent to the first level sub problems where the calculations from p.2a till 3b are repeated and so on.

7. MATLAB CALCULATION OF THE PROBLEM FOR OPTIMAL CONTROL OF THE YEAST FED-BATCH FERMENTATION

The data for the fed-batch fermentation process is taken from the M-Tech dissertation of Njodzi, [9].
The values of the model parameters and the coefficients representing the polynomial functions of the kinetic parameters are obtained on the basis of parameter estimation techniques as follows:

Name	Notation in the text	Value
Initial values of control variable		
- Temperature	T	16
- PH	pH	4.8
- DO_2	DO_2	37
- Flow rate	F	0
Steps of the gradient procedures for state variables	α_x	0.001
	α_s	0.001
	α_p	0.001
	α_v	0.001
	$\alpha_{\lambda x}$	0.001
	$\alpha_{\delta x}$	0.001
Conjugate variable	λ_x	0.001
	λ_s	0.005
	λ_p	0.002
	λ_v	0.003
Interconnections in time domain	δ_x	10.0
	δ_s	138
	δ_p	25.0
	δ_v	2.0
Coefficients	$a_1 - a_7$	0.0004;0.0001;0.0002;0.0001;0.0001; 0.00002;0.00002
	$b_1 - b_7$	4; 0.5; 0.13; 0.13; 0.013; 0.014;0.013
	$c_1 - c_7$	0.0002;0.0001;0.000021;0.00021; 0.000211;0.0000211;0.0002
	$d_1 - d_7$	0.0360;0.008;0.0030.005;0.005; 0.001;0.0002
	K_d	0.00008
	m	0.11
	n	0.52
	s_i	266.6
	p^*	50
	Δt	2.5
Initial values of state variables	x(0)	1.83
	s(0)	150.0
	p(0)	2.1
	v(0)	2.1
Penalty coefficients	μ_x	0.0010
	μ_s	0.0010
	μ_p	0.0010
	μ_v	0.0010
Min and max of state variables	x_{max}	55
	x_{min}	0.83
	s_{max}	300
	s_{min}	20
	p_{max}	70
	p_{min}	2.0
	v_{max}	4
	v_{min}	2
Min and max of control variables	T_{max}	20
	T_{min}	10

	pH_{max}	7
	pH_{min}	4
	DO_{2max}	60
	DO_{2min}	10
	F_{max}	5.0
	F_{min}	0

Table 1: Values of the parameters and coefficient

The number of steps K = 50, the number of iterations on the first and second levels are M1 = 10, M2 = 20. The errors for completing of the calculations on every level are selected to be 0.001.

Matlab program is developed to solve the problem for optimal control in the two level structures. The initial trajectories of state variables are given on Fig 2.The optimal trajectories of the state variables and control variables are given on Fig 3, Fig 4 and Fig 5 respectively.

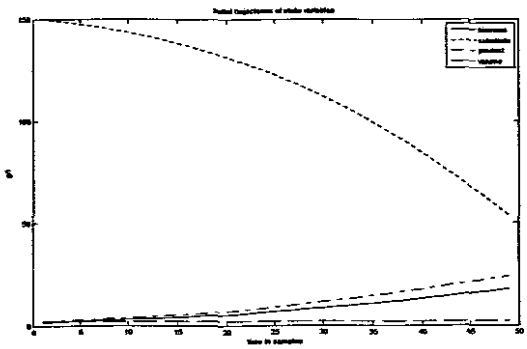


Figure 2: Initial trajectories of the state variables

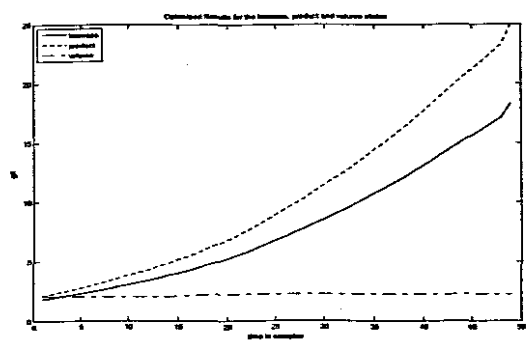


Figure 3: Optimal trajectories of biomass product and volume

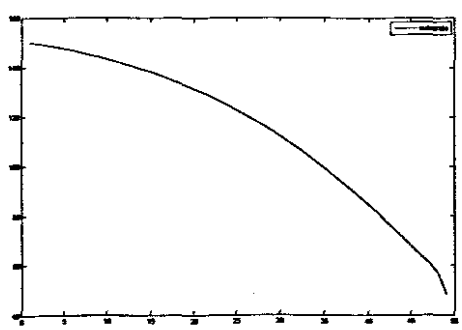


Figure 4: Optimal trajectories of substrate

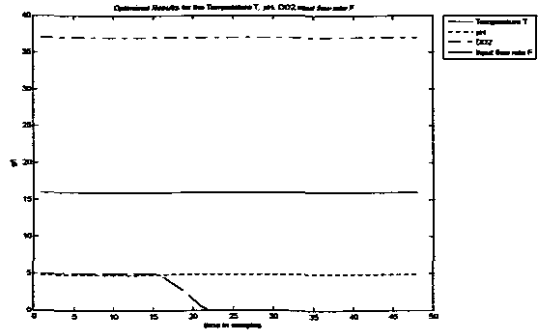


Figure 5: Optimal trajectories of the control variables

The optimal trajectories of the conjugate variables are given on Fig 6.

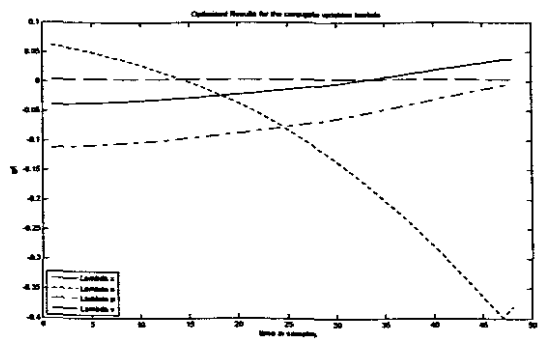


Figure 6: Optimal trajectories of the conjugate variables

8. CONCLUSION

The problem that is considered and solved in the paper is connected with the process behavior optimization based on the existing dependencies between the chemical (environmental) process variables like temperature pH, Dissolved oxygen (DO) and the biological variables (concentration of biomass substrate and product) [10].

The optimal operation of the process or yield of the yeast is based on some criteria for the production of biomass, and some constraints over minimal and maximal values of the variables. Decomposition method to solve the optimal control problem is developed on the bases of an augmented functional of Lagrange and decomposition in time domain. Algorithm of the method and program in Matlab are described.

The convergence of the calculations towards the optimal solution is fast, but depends on the selection of the steps of the gradient procedures and initial values of the coordinating variables.

The Matlab programs perform the calculation in a sequential way. The applied decomposition of the problem solution allows parallel performance of the calculation. This problem is under investigation at the present and will be described in a next publication.

The developed Matlab program is a part of a LabVIEW system for automatic control of fed-batch fermentation

process [11], developed for a fermentor. Biostat®C at the Department of Chemical Engineering, UCT.

9. ACNOWLEDGEMENTS

The research was supported by the National Research Foundation under a grant *2052418*.

10. REFERENCES

[1] POMERLEAU Y., PERRIER M., BOURQUE D., 1995: Dynamics and control of the fed-batch production of poly – β – Hydroxybutyrate by Methyl bacterium extorquens

[2] MICHAEL L. SHULER, FIKRET KARGI, 1992: Bioprocess Engineering Basic Concepts

[3] HIGGIN, J. 1984. Process Control No Longer Separate from Simulation Design. Chemical Engineering News, vol. 62, No. 14, pp. 286-291.

[4] FLAUS J.M., PONS M.N., CHERUY A., and ENGASSER J.M., 1989: Adaptive algorithm for estimation and control of fed-batch ioprocess, Prectice Hall, NY.

[5] SKOGESTAK, S.: HALVORSEN, I: LARSSON, T & GOVATSMSRK, M.1999: Plant-Wide Control: The Search for the self-optimizing Control Structure. Proceedings of the 14th *IFAC World Congress, China, pp. 325-330.*

[6] TAMURA, H. 1975. Decentralized Optimization for Distributed Lag Models of Discrete Systems, Automatica, vol. 11, pp. 593-602.

[7] SING. M. A.TITLI., 1978: Systems: Modelling, Optimization and Control. New York, Pergamon Press.

[8] LILIANA KRZYSTEK & STANISLAW LEDAKOWICZ, 1997: Yield and Maintenance Coefficients in S. Cerevisiae Cultures.

[9] NJODZI Z., 2001: *Studies on the fed-batch propagation of brewer's yeast in high gravity wort,* MSC dissertation, UCT, Cape Town.

[10] NIKOLOVA N., E. NIKOLOV (2004), Fractional Robust Control System with Internal Model, In: Proc. of International Conference AUTOMATICS AND INFORMATICS'04 Sofia, October 6-8, 2004, ISSN 954-9641-39-2, Vol. 1, © 2004 Union of Automation and Informatics, pp. 171-174.

[11] NCEDO S. MKONDWENI, 2002: *Modeling and optimal control of fed-batch fermentation process for the production of yeast,* Department of Electrical Engineering, Faculty of Engineering, Peninsula Technikon, CT, SA.